

PDF テンプレートエンジン

Field Reports for Linux

ユーザーズ・マニュアル

第 2.0.0 版

2022 年 9 月 7

合同会社 フィールドワークス

Field Works, LLC.

まえがき

本書では、PDF テンプレートエンジン Field Reports（以降、Field Reports と表記します）のインストール手順、Field Reports を利用したプログラムの作成手順および作成上の注意事項について説明します。また、レンダリングパラメータの書式および API について解説します。

販売と保守について

ライセンスのご購入

Field Reports のライセンスのご購入は、以下 Web サイトよりお願いします。

<https://www.field-works.co.jp/>

エラーや不具合

エラーや不具合を発見した場合、あるいは改善要望などがございましたら、下記までご連絡ください。

support@field-works.co.jp

ご注意

本書について

1. 本書の内容の一部または全部を無断で転載することはお断りします。
2. 本書の内容は、将来予告なしに変更することがあります。
3. 本書の作成にあたっては正確な記述に努めましたが、本書に基づく運用結果について、合同会社フィールドワークスは責任を負いかねますのでご了承ください。

版権について

すべての権利は、合同会社フィールドワークスに属しています。書面による同意なしに本書の内容を複製・改変および翻訳することを禁じます。

Copyright © 2011–2022 Field Works, LLC All rights reserved.

商標について

- Adobe, Acrobat, Adobe PDF, Adobe Reader は、Adobe Systems Incorporated の登録商標です。
- macOS は、Apple Inc. の登録商標です。
- Microsoft, Windows, Microsoft .NET, Visual Studio, Microsoft Word, Excel は、Microsoft Corporation の登録商標です。
- Linux は、Linus Torvalds 氏の登録商標です。
- その他、本書に掲載されている会社名・製品名は、各社の商標または登録商標です。
- 本書では、® および™ を明記しておりません。

参考文献

1. Adobe Systems. (2004). *PDF Reference fifth edition: Adobe Portable Document Format Version 1.6*.
2. JIS X 4051 (2004) 『日本語文書の組版方法』
3. W3C. (2012). 『日本語組版処理の要件（日本語版）』 東京電機大学出版局
4. W3C. (2018). *Scalable Vector Graphics (SVG) 2*

改版履歴

2021 年 8 月 2 日	第 2.0.0a1 版	α 版リリース
2021 年 11 月 16 日	第 2.0.0b1 版	β 版リリース
2021 年 12 月 10 日	第 2.0.0b2 版	parse API を追加 action 属性等を追加
2022 年 3 月 3 日	第 2.0.0rc 版	可変テーブルに関する記述を追加
2022 年 5 月 23 日	第 2.0.0b3 版	新方式言語 Bridge を導入
2022 年 7 月 6 日	第 2.0.0rc4 版	コマンドライン API を修正
2022 年 8 月 10 日	第 2.0.0rc7 版	Unicode 正規化を廃止
2022 年 9 月 7 日	第 2.0.0rc8 版	Unicode → GID 変換結果を返却

目次

第 1 章	はじめに	1
1.1	Field Reports とは	1
1.1.1	主な特長	1
1.1.2	2.0 版で追加された機能	2
1.1.3	動作環境	2
1.1.4	制限事項	2
1.2	PDF 帳票作成手順	3
1.2.1	PDF テンプレートの作成	3
1.2.2	レンダリングパラメータの作成	6
1.2.3	PDF 帳票の生成	7
第 2 章	インストール手順	8
2.1	インストールの概要	8
2.1.1	Field Reports の構成	8
2.1.2	インストール媒体のファイル構成	9
2.2	Field Reports 本体のインストール	10
2.2.1	参照ライブラリのインストール	10
2.2.2	インストーラの実行	10
2.2.3	動作確認	11
2.2.4	初期設定ファイルについて	11
2.2.5	アンインストール	12
2.2.6	クラウドサービスでのご利用	12
2.3	言語 Bridge のインストール	13
2.3.1	連携手段の選択	13
2.3.2	Python Bridge	13
2.3.3	Ruby Bridge	14
2.3.4	PHP Bridge	15
2.3.5	Java VM Bridge	17
2.3.6	.NET Bridge	19
2.4	ライセンス認証	21
2.4.1	年間ライセンスキー	21
2.4.2	ライセンスキーの登録	21

第 3 章	帳票定義	22
3.1	帳票定義の概要	22
3.1.1	PDF テンプレート	23
3.1.2	レンダリングパラメータ	25
3.2	テンプレート要素	26
3.2.1	単票の場合のテンプレート指定	26
3.2.2	複合帳票の場合のテンプレート指定	26
3.2.3	連続帳票の場合のテンプレート指定	27
3.2.4	複合帳票+連続帳票でのテンプレート指定	28
3.2.5	条件付きテンプレート選択	29
3.3	コンテキスト要素	31
3.3.1	単票でのフィールド値の指定	31
3.3.2	複合帳票でのフィールド値の指定	32
3.3.3	連続帳票でのフィールド値の指定	33
3.3.4	フィールド属性の指定	33
3.4	スタイル要素	35
3.4.1	スタイル指定の例	35
3.4.2	セレクタとは	36
3.5	リソース要素	38
3.5.1	フォントリソース	38
3.5.2	画像リソース	40
3.5.3	リソース URL 指定について	41
3.6	可変テーブル	42
3.6.1	可変テーブルの構成	42
3.6.2	セル値の指定	44
3.6.3	境界線の指定	45
3.6.4	フィールド属性の指定	46
3.6.5	記述例	48
3.7	リッチテキスト (Professional 版)	53
3.8	拡張漢字の利用	56
3.8.1	追加面に格納された Unicode 文字の指定	56
3.8.2	異体字セレクタによる異体字の指定	58
3.8.3	グリフ直接指定	59
第 4 章	帳票生成	62
4.1	帳票生成処理の概要	62
4.1.1	テーブル分割処理	62
4.1.2	レンダリング処理	62
4.2	テーブル分割処理の詳細	65
4.3	レンダリング処理の詳細	67

4.3.1	テキストフィールドの外観生成	67
4.3.2	ボタン（画像）フィールドの外観生成	68
4.3.3	境界線の外観生成	69
4.3.4	回転角度	70
4.3.5	座標変換	70
4.3.6	透明度	71
4.3.7	ブレンドモード	72
第 5 章	レンダリングパラメータ	74
5.1	基本データ	74
5.1.1	JSON で記述する場合	74
5.1.2	Python から利用する場合	75
5.1.3	Ruby から利用する場合	75
5.1.4	PHP から利用する場合	76
5.2	共通データ構造	76
5.2.1	長さ	76
5.2.2	比率	77
5.2.3	日付／時刻	77
5.2.4	色	77
5.2.5	URL	78
5.3	セレクタ文字列	79
5.4	レンダリング辞書	81
5.5	template 要素	82
5.5.1	名前空間	82
5.5.2	PDF 要素	83
5.6	resources 辞書	85
5.6.1	font 要素	85
5.6.2	image リソース辞書	89
5.7	context 要素	90
5.8	style リスト	91
5.9	filed 要素	91
5.9.1	フィールド属性	92
5.9.2	共通フィールド属性	92
5.9.3	コンテンツ・スタイル属性	95
5.9.4	境界線スタイル属性	95
5.9.5	テキスト・バリュー属性	98
5.9.6	テキスト・スタイル属性	103
5.9.7	テキスト・レイアウト属性	106
5.9.8	テキスト改行属性	109
5.9.9	ボタンフィールド属性	111

5.9.10 可変テーブル属性	113
5.10 property 辞書	114
5.10.1 docinfo 辞書	114
5.10.2 metadata	115
5.10.3 encryption 辞書	115
5.10.4 viewer-preferences 辞書	116
5.11 環境変数	117
5.11.1 ユーザー定義環境変数	117
5.11.2 システム定義環境変数	118
5.12 settings 辞書	118
第 6 章 リッチテキスト (Professional 版)	119
6.1 XML 要素	119
6.1.1 文書構造	120
6.1.2 body 要素	120
6.1.3 p 要素	120
6.1.4 span 要素	121
6.1.5 br 要素	121
6.1.6 ruby 要素	121
6.1.7 rt 要素	122
6.1.8 img 要素	122
6.1.9 shape 要素	124
6.2 属性	125
6.2.1 style 属性	125
6.2.2 長さの指定	131
6.2.3 比率の指定	131
6.2.4 色の指定	132
第 7 章 図形要素	133
7.1 XML 要素	133
7.1.1 記述方法	133
7.1.2 line 要素	134
7.1.3 rect 要素	135
7.1.4 circle 要素	135
7.1.5 ellipse 要素	136
7.1.6 polyline 要素	136
7.1.7 polygon 要素	136
7.1.8 path 要素	137
7.1.9 g 要素	137
7.2 属性	137

7.2.1	transform 属性	137
7.2.2	style 属性	138
7.2.3	長さの指定	138
7.2.4	色の指定	139
第 8 章	API リファレンス	140
8.1	Python Bridge API	140
8.1.1	Bridge クラス	140
8.1.2	Proxy インターフェース	141
8.1.3	ExeProxy クラス	142
8.1.4	HttpProxy クラス	142
8.1.5	サンプルプログラム	143
8.2	Ruby Bridge API	143
8.2.1	FieldReports::Bridge クラス	143
8.2.2	FieldReports::Proxy インターフェース	144
8.2.3	ExeProxy クラス	145
8.2.4	HttpProxy クラス	145
8.2.5	ReportsError クラス	145
8.2.6	サンプルプログラム	146
8.3	PHP Bridge API	146
8.3.1	FieldReports\Bridge クラス	146
8.3.2	FieldReports\Proxy インターフェース	147
8.3.3	ExeProxy クラス	148
8.3.4	HttpProxy クラス	148
8.3.5	ReportsException クラス	148
8.3.6	サンプルプログラム	149
8.4	Java VM Bridge API	149
8.4.1	jp.co.field_works.field_reports.Bridge クラス	149
8.4.2	jp.co.field_works.field_reports.Proxy インターフェース	150
8.4.3	jp.co.field_works.field_reports.ExeProxy クラス	151
8.4.4	jp.co.field_works.field_reports.HttpProxy クラス	151
8.4.5	ReportsException クラス	151
8.4.6	サンプルプログラム	152
8.5	.NET Bridge API	152
8.5.1	FieldWorks.FieldReports.Bridge クラス	152
8.5.2	FieldWorks.FieldReports.IProxy インターフェース	154
8.5.3	FieldWorks.FieldReports.ExeProxy クラス	155
8.5.4	FieldWorks.FieldReports.HttpProxy クラス	155
8.5.5	ReportsException クラス	155
8.6	コマンドライン I/F	156

8.7	Web API	157
8.7.1	/version	157
8.7.2	/render	158
8.7.3	/parse	159
付録 A	利用ライブラリ	161
付録 B	文字参照	162
B.1	Lantin 1 Characters	162
B.2	Special Characters	165
B.3	Symbols	166
付録 C	1.5 版からの変更点	169
C.1	フィールド辞書	169
C.1.1	text-align と vertical-align	169
C.1.2	padding	169
C.1.3	normalize	169
C.2	API	170
C.2.1	言語 Bridge	170
C.2.2	コマンドライン I/F	170
C.2.3	低レベル I/F	170

第 1 章

はじめに

1.1 Field Reports とは

マルチプラットフォーム／マルチ言語対応の PDF テンプレートエンジンです。

固定デザインの下絵 (PDF テンプレート) の上に可変データ (レンダリングパラメータ) を重畠した PDF ドキュメント (PDF 帳票) を容易に生成することができます。

業務システムでの帳票出力をはじめとして、チラシ・パンフレット・ダイレクトメール (DM) 等のバリアブル印刷など、さまざまな用途において柔軟にお使いいただくことができます。

1.1.1 主な特長

- マルチプラットフォーム対応

Linux, Windows, macOS の各プラットフォーム上で動作します。

- マルチ言語対応

Python, Ruby, PHP ならびに Java VM, .NET プラットフォーム上で動作するプログラミング言語 (Java, Scala, C#, VB.NET 等) をサポートします。

- クラウド環境対応

サーバーモードで起動することにより、Web API サーバーとして動作させることができます。

クラウド環境のシステムにおいて、帳票サーバーを容易に構成することができます。

- テンプレート方式

(フォーム) フィールドを配置した PDF をテンプレートとして利用します。テキスト・画像等の可変データは JSON ベースのレンダリングパラメータとして記述します。Field Reports は、PDF テンプレートとレンダリングパラメータを合成 (レンダリング) して、PDF 帳票を生成します。

- オフィスソフトと Adobe Acrobat を用いた帳票設計

PDF テンプレートの作成には、Excel, Word 等のオフィスソフトと Adobe Acrobat を利用します。普段ご利用のオフィスソフトを利用して、イメージどおりの帳票を設計できます。

- 日本語組版機能

JIS X 4051 にもとづく自動組版エンジンを実装しています。

- PDF1.6 準拠

PDF version 1.6 に準拠した PDF 帳票を出力します。

暗号化、データ圧縮、埋め込みフォントなどに対応します。

1.1.2 2.0 版で追加された機能

- リッチテキスト（Professional 版）
テキストに文字単位で装飾を付けることができます。
- 可変テーブル
行数可変のテーブルを作成できます。
- 図形要素
SVG サブセット形式で、ベクトル図形を記述できます。

1.1.3 動作環境

Field Reports for Linux は、表 1.1 の環境で動作します。

表 1.1: 動作環境

項目	条件
対象 OS	Linux Kernel: 5.3 以上 アーキテクチャ: x86_64 依存ライブラリ : glibc 2.17 以上, GMP 6, libev 4 文字コード (ロケール) : ja_JP.UTF-8
必要なソフトウェア	Python 3.7 以上 Ruby 2.7 以上 PHP 7.4 以上 Java SE 8 以上 .NET Core SDK 3.1 以上 Adobe Acrobat Pro ^{*1}
ハードウェア	CPU : お使いの Linux ディストリビューションが推奨する環境以上 メモリ : 同上 HDD : 10MB 以上

その他の環境での動作報告については、弊社 Web サイト (<https://www.field-works.co.jp/>) でご確認ください。

1.1.4 制限事項

- マルチスレッド環境下での並列動作はサポートしていません。
並列処理が必要な場合は、マルチプロセス環境でご利用ください。
- 生成可能な PDF のページ数・ファイルサイズの上限は、実行環境で利用可能なメモリのサイズに依存します。

^{*1} PDF にフィールドを配置する必要があるため、Pro 版が必要になります。

1.2 PDF 帳票作成手順

写真付き料理レシピの帳票作成を題材として、PDF 帳票を生成するまでの手順を説明します。

1.2.1 PDF テンプレートの作成

最初に、Microsoft Word, Excel などのオフィスソフトを使って、PDF テンプレートの下絵となる文書を作成します。その文書を、Adobe Acrobat などのツールを用いて、PDF ファイルに変換します。

今回は、オフィスソフトとして OpenOffice.org の Calc を使用しましたので（図 1.1），PDF ファイルへの変換まで OpenOffice.org で行うことができます（図 1.2）。

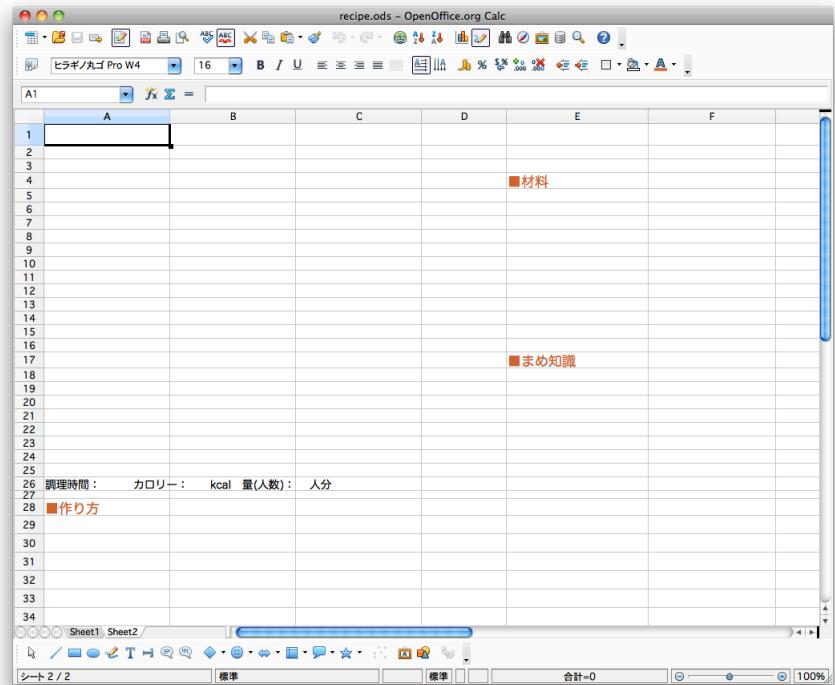


図 1.1: 下絵の作成

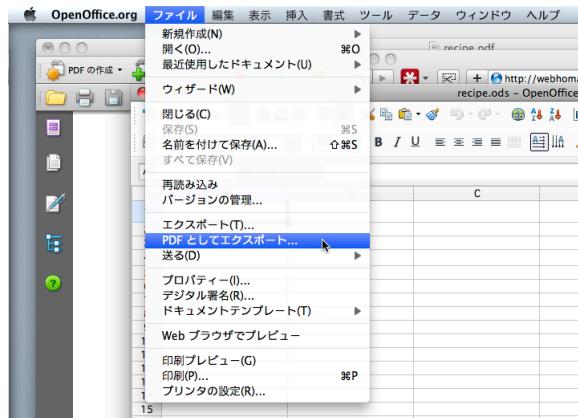


図 1.2: PDF への変換

次に、作成した PDF ファイルを Adobe Acrobat などの PDF 編集が可能なアプリケーションで開き、フィールドを配置します。フィールド名は、後でフィールドを参照する際に使用しますので、わかりやすい名前をつけてください。フォント・フォントサイズ・表示色などの表示属性もこの時点では設定します（帳票生成時に動的に変更することも可能です）。

図 1.3 は、Adobe Acrobat を使ってフィールドの配置を行っている様子です。テーブル形式のフィールドを作成する際には、「複数のフィールドを配置...」または「複数のコピーを作成...」を使用すると便利です。

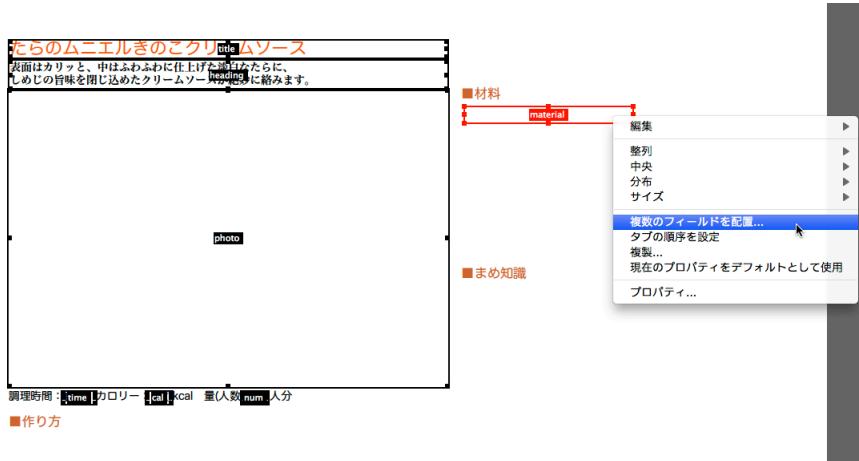


図 1.3: フィールドの配置

すべてのフィールドの配置が終わったら、保存します（図 1.4）。ここでは、ファイル名を “recipe.pdf” としました。

ここまでで、PDF テンプレートの作成は完了です。

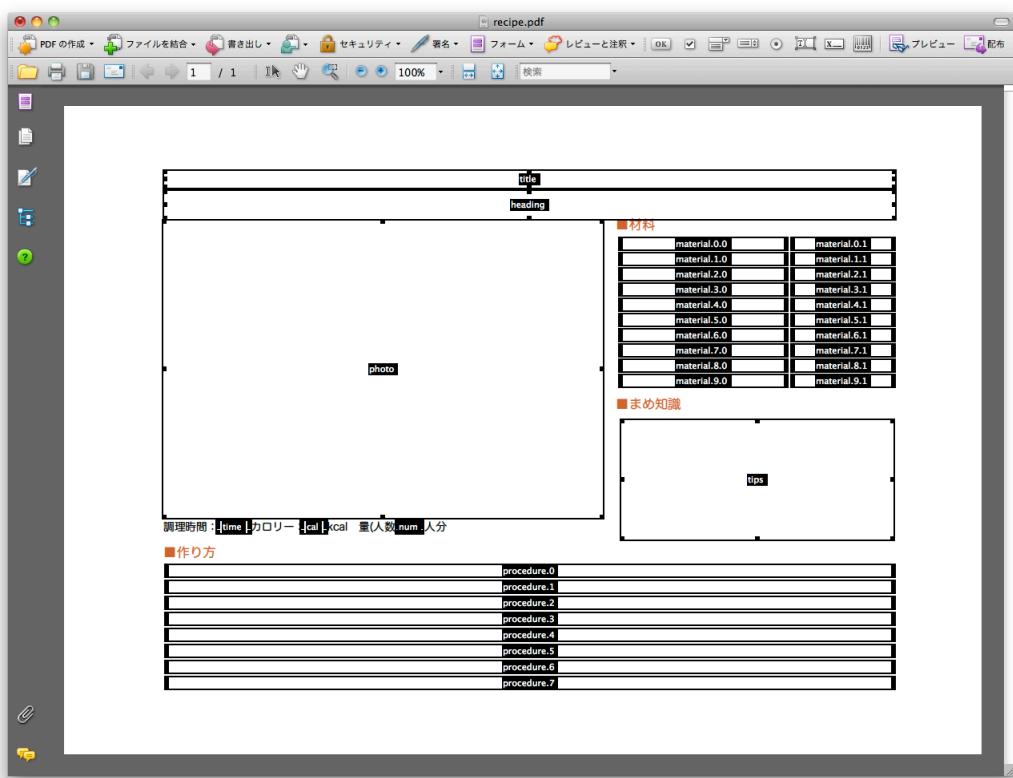


図 1.4: フィールドの配置が完了した状態

1.2.2 レンダリングパラメータの作成

レンダリングパラメータを JSON 形式のファイルとして作成します。

作成したレンダリングパラメータは，“recipe.json” というファイル名で保存します。エンコーディングは、UTF-8 とします。画像ファイル “meuniere_photo.jpg” もレンダリングパラメータと同一のディレクトリに配置しておきます。

```
{
  "template": "./recipe.pdf",
  "context": {
    "title": {"value": "たらのムニエルきのこクリームソース", "color": [204, 102, 51]},
    "heading": "表面はカリッと、中はふわふわに仕上げた淡白なたらに、\nしめじの旨味を閉じ込めたクリー
    ↵ ムソースが絶妙に絡みます。",
    "photo": {"icon": "./recipe_photo.jpg"},
    "time": "15分",
    "cal": 310,
    "num": "2",
    "material": [
      ["たら（切身）", "2枚"],
      ["しめじ", "1パック"],
      ["小麦粉（強力粉）", "適量"],
      ["生クリーム", "50cc"],
      ["卵黄", "1個"],
      ["バター", "20g"],
      ["塩", "適量"],
      ["白ワイン", "大さじ2"],
      ["チャービル", "適量"]
    ],
    "procedure": [
      "(1)たらの表面に塩をふり、小麦粉（強力粉）をまぶし、バター50gを入れフライパンで、皮の方から焼く
      ↵ 。",
      "(2)焼き上がったら皿に写し、フライパンの余分な油をとる。",
      "(3)残りのバター・白ワイン・しめじを炒め、生クリーム・卵黄を加え軽く火を通す。",
      "(4)(2)に(3)をかけ、チャービルを飾る。"
    ],
    "tips": "昔、風車を回して小麦粉を作っている人をフランス語で「ムニエ」 と呼んでいました。小麦粉を使
    ↵ った料理「ムニエル」は、この「ムニエ」が由来しているそうです。"
  }
}
```

コード 1.1: recipe.json

1.2.3 PDF 帳票の生成

reports コマンドの実行

“recipe.pdf”と“recipe.json”が存在するディレクトリで、以下のコマンドを実行します。生成された PDF 帳票は、“meuniere.pdf”に保存されます。

```
$ reports render recipe.json meuniere.pdf
```

完成イメージ

完成した PDF 帳票のイメージを図 1.5 に示します。



図 1.5: 生成された PDF 帳票 (イメージ)

第 2 章

インストール手順

2.1 インストールの概要

2.1.1 Field Reports の構成

Field Reports は、表 2.1 の要素により構成されています。

表 2.1: ソフトウェアの構成

分類	内訳
Field Reports 本体	コマンドラインプログラム
言語 Bridge	Python Bridge
	Ruby Bridge
	PHP Bridge
	Java VM Bridge
	.NET Bridge

最初に Field Reports 本体のインストールを行い、次に必要なプログラミング言語用の Bridge をインストールします。

2.1.2 インストール媒体のファイル構成

インストール媒体を tar コマンドにより展開します。

```
# tar xvzf reports-2.x.x-linux.tar.gz
```

展開したインストール媒体は、以下のファイル構成となっています。

```
reports-2.x.x-linux
├── dotnet/
├── etc
│   └── reports.conf
├── examples/
├── jvm/
├── php/
├── python/
├── ruby/
├── CHANGELOG.md
├── LICENSE.md
├── README.md
├── license_lgpl.txt
├── reports-2.x.x-x86_64.rpm
├── reports-2.x.x-1_amd64.deb
└── users-man.pdf
```

- “2.x.x” または “2.x” は、Field Reports のバージョン番号を示します（以下同様）。
- dotnet, jvm, php, python, ruby フォルダには、各言語 Bridge のインストーラまたはソースが格納されています。
- README ファイルやサンプルプログラム等のテキストファイルの文字コードは UTF-8 です。

2.2 Field Reports 本体のインストール

2.2.1 参照ライブラリのインストール

Field Reports は、表 2.2 に示すライブラリを動的リンクしています。

表 2.2: 動的リンクライブラリ

名称	ファイル名
GNU C Library	libc.so.6, libm.so.6, librt.so.1, libdl.so.2, libpthread.so.0
GMP	libgmp.so.10
libev	libev.so.4

未導入のライブラリがある場合は、パッケージマネージャーを利用するかソースからビルドするなどして、インストールしてください。

■RedHat(CentOS) 系

```
# yum install gmp  
# yum install libev
```

■Debian(Ubuntu) 系

```
# apt install libgmp10  
# apt install libev4
```

2.2.2 インストーラの実行

OS のアーキテクチャに対応した RPM ファイルまたは DEB ファイルを使ってインストールしてください。インストールには、管理者権限が必要です。

■RedHat(CentOS) 系

```
# rpm -ivh reports-2.x.x-<アーキテクチャ>.rpm
```

■Debian(Ubuntu) 系

```
# dpkg -i reports_2.x.x-1_<アーキテクチャ>.deb
```

■インストール結果

“/usr/bin”にコマンドライン・プログラムがインストールされます。

```
/usr
└── bin
    └── reports
```

注意事項

- バージョンアップの場合は、最初に [2.2.5](#) の手順にしたがって、アンインストールしてください。

2.2.3 動作確認

コマンドライン・プログラムが起動できることを確認してください。

```
$ reports
Field Reports Trial 2.x.x [linux/x86_64]
usage: reports <subcommand> [options] [args]
Type 'reports <subcommand> ----help' for help on a specific subcommand.
```

Available subcommands:

reports version	show Field Reports version
reports render	render PDF report file
reports parse	parse PDF and show structure
reports font	probe font file
reports check	check initial config file
reports server	start as HTTP server

Copyright 2011-20xx Field Works, LLC

<https://www.field-works.co.jp/>

“cannot open shared object file”のようなエラーが表示される場合は、参照している動的ライブラリの存在を確認してください。

```
$ ldd $(which reports)
```

2.2.4 初期設定ファイルについて

初期設定ファイルを所定の場所に置いて、コンピュータ内で共通のパラメータを記述しておくことができます。初期設定ファイルに記述された値は、個々のPDF帳票を生成する際に指定するレンダリングパラメータ

の初期値として利用されます。

初期設定ファイルには主にライセンスキーを登録するための settings 辞書を記述しますが、style リスト・resources 辞書・environ 辞書・property 辞書を記述することも可能です（context 要素は記述されていても無視します）。

初期設定ファイルのデフォルトのパス名は、”/etc/reports.conf” です。環境変数 REPORTS_CONFIG を設定することにより、初期設定ファイルのパス名を変更することができます。

Web サーバから Field Reports を利用する場合は、Web サーバのプロセスから初期設定ファイルが読み込めるように、適切な権限を設定してください。

2.2.5 アンインストール

管理者権限で以下のコマンドを実行してください。

■RedHat(CentOS) 系

```
# rpm -e reports
```

■Debian(Ubuntu) 系

```
# dpkg -r reports
```

2.2.6 クラウドサービスでのご利用

実行ファイルの配置

PaaS 等の通常の方法でのインストールができない環境下で Field Reports を利用される場合は、実行ファイルを抽出して実行環境にアップロードしてください。

また、クラウドサービスで用意されている手段で、参照ライブラリ（[2.2.1](#)）をインストールしてください。

初期設定ファイル

所定の場所に初期設定ファイルが置けない場合は、環境変数 REPORTS_CONFIG で場所を指定するか、帳票生成時のレンダリングパラメータで毎回ライセンスキー情報を受け渡してください。

2.3 言語 Bridge のインストール

2.3.1 連携手段の選択

本モジュールと Field Reports 本体との連携方法として、以下の 2 種類があります。システム構成に応じて、適切な連携方法を選択してください。

■コマンド呼び出しによる連携

- Field Reports 本体と言語 Bridge を同一マシンに配置します。
- パスが通る場所に reports コマンドを置くか、reports コマンドのパスを API に渡してください。

■HTTP 通信による連携

- Field Reports 本体をリモートマシンに配置することができます。
- Field Reports は、サーバーモードで常駐起動させてください ('reports server')。
- サーバーモードで使用するポート番号（既定値：'50080'）の通信を許可してください。

2.3.2 Python Bridge

Python 拡張モジュールのインストール

■インストール媒体からのインストール

pip コマンドを利用して、インストール媒体の wheel ファイルをインストールしてください。

```
$ pip3 install field_reports-2.x.x-py3-none-any.whl
```

■GitHub からのインストール

GitHub に登録されているソースコードからインストールする場合は、以下のコマンドを実行してください。

```
$ pip3 install git+https://github.com/field-works/python-bridge.git@2.x.x
```

動作確認

■コマンド連携時

以下のコマンドを実行してください。

```
$ python3
>>> import field_reports
>>> reports = field_reports.Bridge.create_proxy("exec:/usr/local/bin/reports")
>>> reports.version()
'2.x.x'
>>> reports.render({})
```

```
b"%PDF-1.6\n%\x80\x81\x82\x83\n..."
```

- 動作環境に応じて、'create_proxy()' に与えるパスを適宜変更してください。

■HTTP 通携時

Field Reports をサーバーモードで起動してください。

```
$ reports server -l3
```

次に、以下のコマンドを実行してください

```
$ python3
>>> import field_reports
>>> reports = field_reports.Bridge.create_proxy("http://localhost:50080/")
>>> reports.version()
'2.x.x'
>>> reports.render({})
b"%PDF-1.6\n%\x80\x81\x82\x83\n..."
```

- 動作環境に応じて、create_proxy() に与える URL を適宜変更してください。

2.3.3 Ruby Bridge

Ruby 拡張モジュールのインストール

■インストール媒体からのインストール

gem コマンドを利用して、インストール媒体の gem ファイルをインストールしてください。

```
$ gem install field_reports-2.x.x.gem
```

■GitHub からのインストール

GitHub に登録されているソースコードからインストールする場合は、以下のコマンドを実行してください。

```
$ gem install specific_install
$ gem specific_install -l https://github.com/field-works/ruby-bridge.git -b 2.x.x
```

動作確認

■コマンド連携時

以下のコマンドを実行してください。

```
$ irb
> require 'field_reports'
> reports = FieldReports::Bridge.create_proxy("exec:/usr/local/bin/reports")
> reports.version()
=> "2.x.x"
> reports.render({})
=> "%PDF-1.6\n%\x80\x81\x82\x83\n..."
```

- 動作環境に応じて、`create_proxy()` に与えるパスを適宜変更してください。

■HTTP 通携時

Field Reports をサーバーモードで起動してください。

```
$ reports server -l3
```

次に、以下のコマンドを実行してください

```
$ irb
> require 'field_reports'
> reports = FieldReports::Bridge.create_proxy("http://localhost:50080/")
> reports.version()
=> "2.x.x"
> reports.render({})
=> "%PDF-1.6\n%\x80\x81\x82\x83\n..."
```

- 動作環境に応じて、`create_proxy()` に与える URL を適宜変更してください。

2.3.4 PHP Bridge

拡張モジュールのインストール

■ソースファイルからのインストール

ソースファイルを展開して、'field_reports'ディレクトリ配下のソースファイルを任意の格納場所にコピーしてご利用ください。

■Composerを利用してのインストール

Composer から参照可能なモジュールが、GitHub に登録されています。
'composer.json'ファイルを下記のように記述してください。

```
{  
    "require": {  
        "field-reports/php-bridge": "x.x.x"  
    },  
    "repositories": [  
        {  
            "type": "vcs",  
            "url": "https://github.com/field-works/php-bridge.git"  
        }  
    ]  
}
```

コード 2.1: composer.json

以下のコマンドを実行してください。

```
$ composer install
```

※ Token の入力が求められた場合は、個人アクセストークンを作成して入力してください。

<https://github.com/settings/tokens>

動作確認

■コマンド連携時

以下のコマンドを実行してください。

```
$ php -a  
php > require('<ソース格納場所>/field_reports/bridge.php');  
php > $reports = FieldReports\Bridge::create_proxy("exec:/usr/local/bin/reports");  
php > echo $reports->version();  
2.x.x  
php > echo $reports->render([]);  
%PDF-1.6...
```

- 動作環境に応じて、create_proxy() に与えるパスを適宜変更してください。

■HTTP 通携時

Field Reports をサーバーモードで起動してください。

```
$ reports server -13
```

次に、以下のコマンドを実行してください。

```
$ php -a
php > require('<ソース格納場所>/field_reports/bridge.php');
php > $reports = FieldReports\Bridge::create_proxy("http://localhost:50080/");
php > echo $reports->version();
2.x.x
php > echo $reports->render([]);
%PDF-1.6...
```

- 動作環境に応じて、`create_proxy()` に与える URL を適宜変更してください。

2.3.5 Java VM Bridge

パッケージのインストール

■jar ファイルによるインストール

インストール媒体より、jar ファイルを任意の格納場所にコピーしてください。

`field_reports-2.x.x.jar`

jar ファイルの格納場所は、環境変数'CLASSPATH' または実行時引数'-classpath' で指定してください。

■Maven プロジェクトからの利用

Maven から参照可能なインストールモジュールを GitHub で配布しています。

プロジェクトから利用する際には、'pom.xml' ファイルに下記の記述を追加してください。

```

<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/maven-v4_0_0.xsd">
<modelVersion>4.0.0</modelVersion>

<dependencies>
  <dependency>
    <groupId>jp.co.field_works</groupId>
    <artifactId>field_reports</artifactId>
    <version>2.x.x</version>
  </dependency>
</dependencies>

<repositories>
  <repository>
    <id>field_works</id>
    <name>Field Works, LLC repository</name>
    <url>https://raw.githubusercontent.com/field-works/jvm-bridge/repo/</url>
  </repository>
</repositories>

</project>

```

コード 2.2: pom.xml

動作確認

■コマンド連携時

以下のコマンドを実行してください。

```

$ jshell --class-path <jar ファイル格納場所>/field_reports-2.x.x.jar
jshell> import jp.co.field_works.field_reports.*
jshell> jp.co.field_works.field_reports.Proxy reports
      = Bridge.createProxy("exec:/usr/local/bin/reports")
jshell> reports.version()
$3 ==> "2.x.x"
jshell> reports.render("{}")
$4 ==> byte[672] { 37, 80, 68, 70, 45, 49, ...

```

- 動作環境に応じて、`create_proxy()` に与えるパスを適宜変更してください。

■HTTP 通携時

Field Reports をサーバーモードで起動してください。

```
$ reports server -l3
```

次に、以下のコマンドを実行してください

```
$ jshell --class-path <jar ファイル格納場所>/field_reports-2.x.x.jar
jshell> import jp.co.field_works.field_reports.*
jshell> jp.co.field_works.field_reports.Proxy reports
      = Bridge.createProxy("http://localhost:50080/")
jshell> reports.version()
$3 ==> "2.x.x"
jshell> reports.render("{}")
$4 ==> byte[672] { 37, 80, 68, 70, 45, 49, ...
```

- 動作環境に応じて、create_proxy() に与える URL を適宜変更してください。

2.3.6 .NET Bridge

開発環境

.NET SDK もしくは Visual Studio が導入済みであるものとします。

以下、dotnet コマンドからの利用を前提として説明します。

パッケージのインストール

■NuGet からのインストール

本モジュールは、NuGet gallery に登録されています。利用している開発環境に応じた方法でインストールしてください。

```
$ dotnet add package FieldWorks.FieldReports --version 2.x.x
```

動作確認

動作確認用のコンソールプロジェクトを作成します。

```
$ dotnet new console -o test
$ cd test
$ dotnet add package FieldWorks.FieldReports
```

Program.cs を以下のように編集します。

```
using FieldWorks.FieldReports;

var reports = Bridge.CreateProxy();
Console.WriteLine(reports.Version());
Console.WriteLine(System.Text.Encoding.ASCII.GetString(reports.Render("{}")));
```

コード 2.3: Program.cs

■コマンド連携時

以下のコマンドを実行してください。

```
$ REPORTS_PROXY=exec:/usr/local/bin/reports dotnet run
2.x.x
%PDF-1.6
...
```

■HTTP 連携時

Field Reports をサーバーモードで起動してください。

```
$ reports server -l4
```

以下のコマンドを実行してください（動作環境に応じて、URL は変更してください）。

```
$ REPORTS_PROXY=http://localhost:50080/ dotnet run
2.x.x
%PDF-1.6
...
```

2.4 ライセンス認証

シリアル番号とライセンスキーを登録することで、試用版から製品版に移行することができます。

2.4.1 年間ライセンスキー

年間ライセンスを新規にご契約頂いた際にシリアル番号と共に年間ライセンスキーが発行されます。次年度以降の年度更新時には、新しい年間ライセンスキーが発行されます。

プログラム開発時には、契約者所有の開発用コンピュータに登録してお使いいただけます。運用時には、契約いただいた台数の本番稼働用コンピュータに登録してお使いください。

2.4.2 ライセンスキーの登録

初期設定ファイル (2.2.4) に下記の書式でライセンス情報を追加してください（ファイルが存在しない場合は、テキストエディタ等で作成してください）。

```
{  
    "settings": {  
        "serial-number": "<シリアル番号>",  
        "auth-code": "<ライセンスキー>"  
    }  
}
```

コード 2.4: reports.conf

確認のため、コマンドライン・プログラムを実行してください。

```
$ reports  
Field Reports Professional 2.x.x [Linux/x86_64]  
usage: reports <subcommand> [options] [args]  
expiration date: YYYY-MM-DD  
...
```

1行目の表示から「Trial」の文字が消えていれば、ライセンスキーの登録は成功です。
「Professional」部分の表示は、エディションにより異なります。

第3章

帳票定義

3.1 帳票定義の概要

Field Reports では、PDF テンプレートとレンダリングパラメータを用いて生成する PDF 帳票を定義します。PDF テンプレートで固定のデザインを規定し、レンダリングパラメータで可変データを指定します。

図 3.1 に PDF テンプレートとレンダリングパラメータを元に帳票を生成するまでの処理の流れを示します。

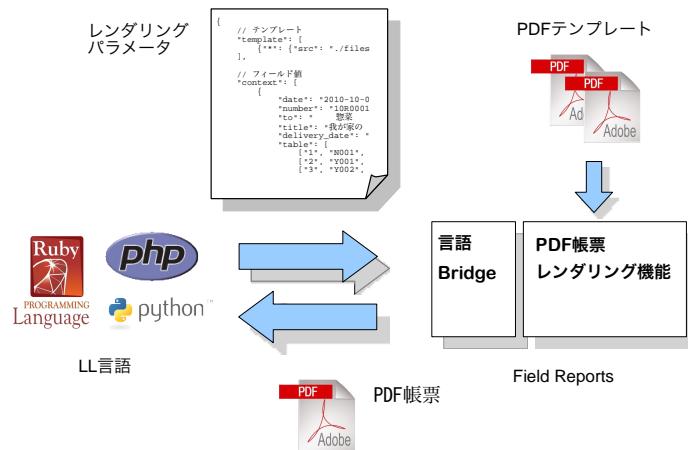


図 3.1: Field Reports 概念図

3.1.1 PDF テンプレート

テキスト・画像を表示したい位置に（フォーム）フィールドを配置した PDF ファイルを PDF テンプレートと呼びます（図 3.2）。

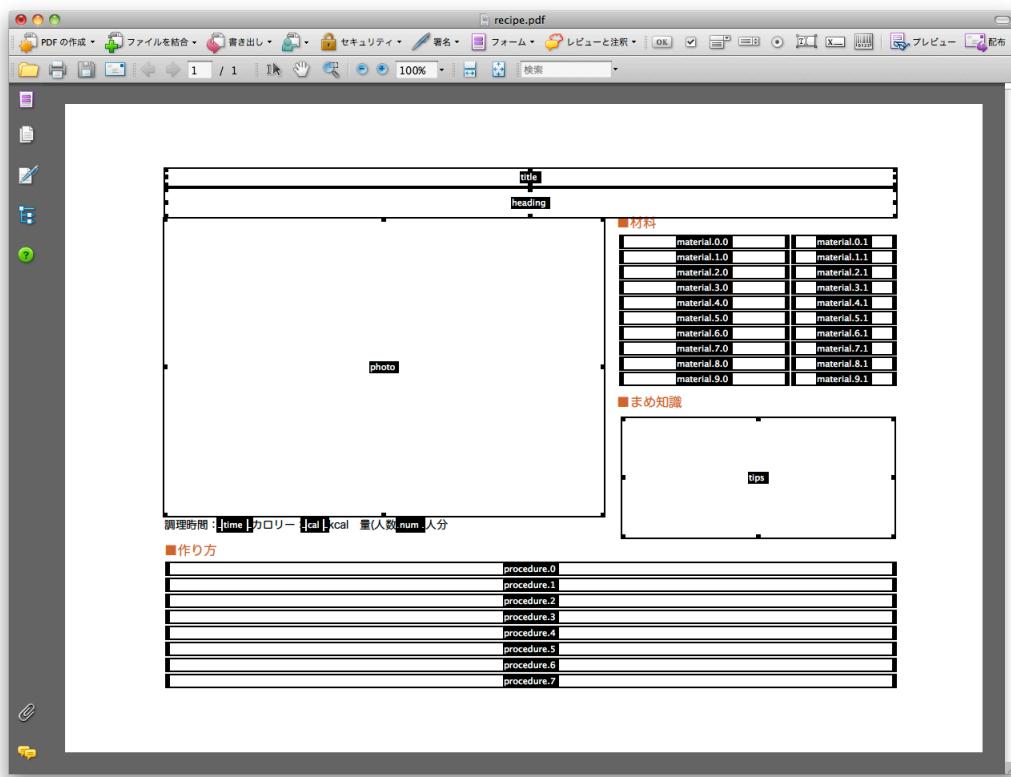


図 3.2: PDF テンプレートの例

フィールドとは

フィールドとは、PDF のページ上に配置された入力エリアです。フォームもしくはフォーム・フィールドとも呼ばれます。本書では、フィールドという呼称で統一しています。

フィールドの本来の用途は、ユーザーにインタラクティブに値を入力させるためのものですが、Field Reports ではテキストや画像を流し込む先のプレースホルダとして利用しています。

フィールド名とは

PDF に配置されたフィールドは「フィールド名」により一意に識別することができます。

フィールド名をピリオドで区切ることで、フィールドの親子関係（階層構造）が表現できます。ルートから末端のフィールド名までをピリオドで区切って並べた形式のフィールド名を「完全修飾フィールド名」と呼び

ます。一方、完全修飾フィールド名の一部を構成するフィールド名を「部分フィールド名」と呼びます。

完全修飾フィールド名 → 部分フィールド名₁ . 部分フィールド名₂ 部分フィールド名_n

テーブル形式フィールド

テーブル形式のデータを配置する際には、テーブルを構成する field 要素の一つ一つに、以下の形式でフィールド名が付けられることを想定しています。

1次元テーブル： テーブル名 . 行番号

2次元テーブル： テーブル名 . 行番号 . 列番号

ここで、行番号・列番号は0始まりの整数です。

Adobe Acrobat では、「複数のフィールドを配置...」または「複数のコピーを作成...」を利用すれば、このような命名規則で規則的に並べたフィールドを一括で作成することができます（図 3.3）。

No	商品コード	品名	数量
table.0.0	table.0.1	table.0.2	table.0.3
table.1.0	table.1.1	table.1.2	table.1.3
table.2.0	table.2.1	table.2.2	table.2.3
table.3.0	table.3.1	table.3.2	table.3.3
table.4.0	table.4.1	table.4.2	table.4.3
table.5.0	table.5.1	table.5.2	table.5.3
table.6.0	table.6.1	table.6.2	table.6.3
table.7.0	table.7.1	table.7.2	table.7.3

図 3.3: テーブル形式フィールド

基本フィールド属性とは

フィールド属性のうち、PDF の仕様に対応するものを基本フィールド属性と呼んでいます。Field Reports で対応している基本フィールド属性の一覧を表 3.1 に示します。

表 3.1: 基本フィールド属性

フィールド属性		テキスト	ボタン
値	テキスト	<input type="radio"/>	-
アイコン	画像	-	<input type="radio"/>
境界線と色	境界線の色・幅・スタイル, 塗りつぶしの色	<input type="radio"/>	<input type="radio"/>
テキスト	フォント・サイズ	<input type="radio"/>	-
オプション	整列, 複数行, リッチテキスト	<input type="radio"/>	-
座標	位置, 幅, 高さ, 向き	<input type="radio"/>	<input type="radio"/>
アクション	JavaScript を実行	<input type="radio"/>	<input type="radio"/>

これら以外の属性（例えば、Adobe Acrobat のフィールドのプロパティダイアログにおける「フォーマット」「検証」「計算」に相当する属性など）は変更することはできません。また、テキスト・ボタン以外の種類のフィールドの属性を変更することもできません。

拡張フィールド属性とは

回転角度・透明度・ブレンドモードなど、Field Reports が独自に拡張したフィールド属性を設定することができます（表 3.2）。

表 3.2: 拡張フィールド属性

フィールド属性		テキスト	ボタン
座標変換	回転角度, 変換行列	<input type="radio"/>	<input type="radio"/>
重ねあわせ	透明度, ブレンドモード	<input type="radio"/>	<input type="radio"/>
余白調整	パディング	<input type="radio"/>	<input type="radio"/>
レイアウト調整	垂直方向整列, 行の高さ,	<input type="radio"/>	-
組版処理	ハイフネーション, 禁則処理, 均等割, 縦組み	<input type="radio"/>	-
拡張漢字	サロゲートペア, 異体字セレクタ, グリフ直接指定	<input type="radio"/>	-
書式指定	数値書式, 日付書式, 文字参照	<input type="radio"/>	-

3.1.2 レンダリングパラメータ

レンダリングパラメータでは、ページの構成要素、フィールド名と可変データの対応、スタイル指定、リソース定義などを記述します。

表 3.3 にレンダリングパラメータの主な構成要素を示します。

表 3.3: レンダリングパラメータの構成要素

要素名	説明
テンプレート	名前空間を定義し、PDF テンプレートと関連付けます。
コンテキスト	フィールド名に設定する属性を 1 対 1 対応で指定します。
スタイル	複数のフィールドに対して、一括してフィールド属性を指定します。
リソース	フォント・画像リソースを定義します。

以降の節では、レンダリングパラメータの各構成要素について説明します。

3.2 テンプレート要素

テンプレート要素では、生成する PDF 帳票のページ構成を定義します。

3.2.1 単票の場合のテンプレート指定

1種類の PDF テンプレートを元に作成される帳票を単票と呼びます。

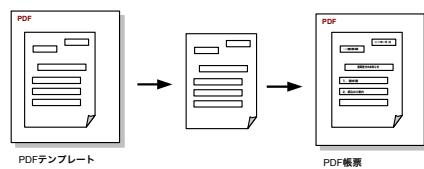


図 3.4: 単票

単票では、`template` 要素の値として、PDF テンプレートのパス名を直接指定します。

```
{  
  "template": "./mitumori.pdf"  
}
```

コード 3.1: 単票 `template` 要素の例

単票ではフィールド名の重複を考慮する必要がないので、名前空間を挿入する必要はありません。元々のフィールド名をそのまま使用します。

3.2.2 複合帳票の場合のテンプレート指定

複数の PDF テンプレートを組み合わせて作成される帳票を複合帳票と呼びます（図 3.5）。

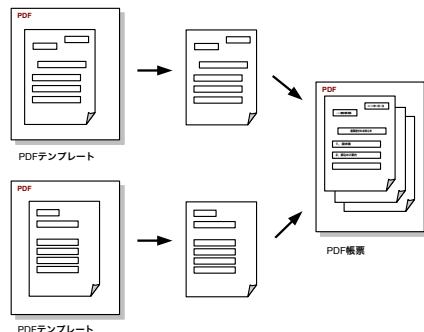


図 3.5: 複合帳票

複合帳票では、PDF テンプレートに任意の部分フィールド名を付けて名前空間を分けます。部分フィールド名をキーにパス名を値とした辞書を作成して、出現順にリスト形式で並べます。

```
{
  "template": [
    {"header": "./hyousi.pdf"},
    {"body": "./mitumori.pdf"}
  ]
}
```

コード 3.2: 複合帳票 template 要素の例

テンプレートに対応付けられた部分フィールド名は、名前空間の先頭に挿入されます（図 3.6）。例えば、a.pdf, b.pdf でそれぞれ「title」という同じ名称のフィールドが定義してあったとすると、名前空間 A と B をそれぞれ挿入することで、「A.title」「B.title」のように区別できるようになります。

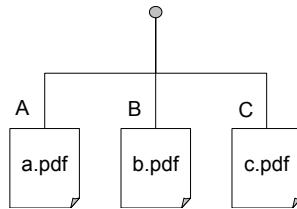


図 3.6: 複合帳票の名前空間

3.2.3 連続帳票の場合のテンプレート指定

テーブル形式のデータの項目数によって、ページ数が変化する帳票を連続帳票と呼びます（図 3.7）。

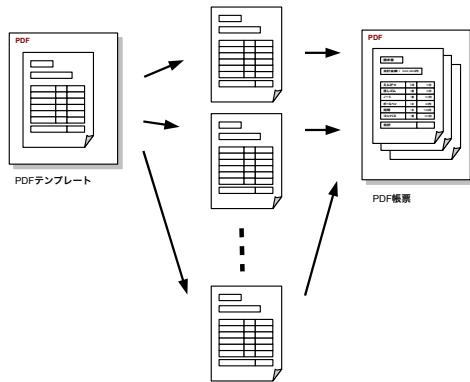


図 3.7: 連続帳票

連続帳票では、テンプレートの名前空間を “*” として、複合帳票と同様に template 要素を記述します。

以下の例では、テンプレートに関する情報を辞書形式で指定しています。rows 属性は、テーブルの最大行数を示しています（可変テーブルでは不要）。

```
{
  "template": [
    {"*": {"src": "./mitumori.pdf", "rows": 10}}
  ]
}
```

コード 3.3: 連続帳票 template 要素の例

連続帳票では、 $0, 1, 2, \dots$ のような 0 始まりの整数の名前空間が暗黙的に定義されますので、それらの名前空間を各ページに割り振ります（図 3.8）。

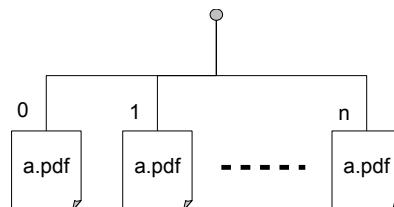


図 3.8: 連続帳票の名前空間

3.2.4 複合帳票 + 連続帳票でのテンプレート指定

複合帳票の一部に連続帳票を含む形式の帳票も作成することができます（図 3.9）。

上記の例では、連続帳票部分の部分フィールド名が “body.*” となり、連続帳票の例で示した “*” より名前空間の階層が 1 段深くなっています（図 3.9）。これは、コンテキスト要素のデータ構造上の都合のために必要なものですが、このような書き方をすることで、複数の連続帳票を組み合わせることも可能になります。

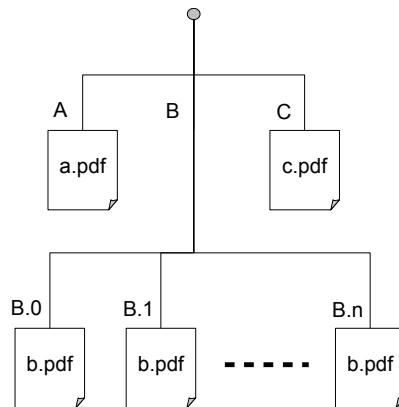


図 3.9: 複合帳票 + 連続帳票の名前空間

```
{
  "template": [
    {"*": {"src": "./mitumori.pdf", "rows": 10}}
  ]
}
```

コード 3.4: 複合帳票+連続帳票の template 要素の例

3.2.5 条件付きテンプレート選択

連続帳票部分では、単に同じデザインのテンプレートを繰り返すだけでなく、条件に応じてテンプレートを切り替えることができます。

例えば、以下のような指定が可能です。

- 最初のページだけ一部デザインが異なるページにする。
- 奇数ページと偶数ページでデザインを変える。

テンプレートの選択条件は、名前空間 “ $0, 1, 2, \dots$ ” に対する部分セレクタとして記述されます。図 3.10 は、名前空間が偶数の時に選択されるテンプレートが “a.pdf”，偶数の時に選択されるテンプレートが “b.pdf” の時の動作イメージです。

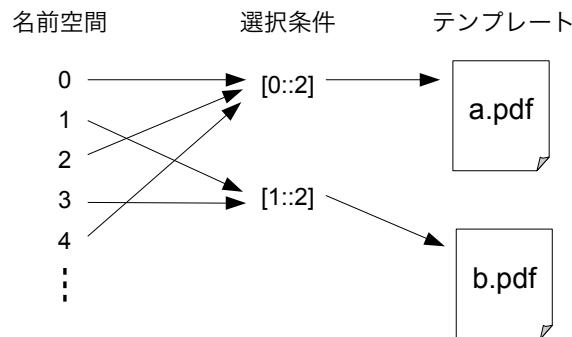


図 3.10: 条件付きテンプレート選択のイメージ

```
{  
    // 偶数ページでa.pdf, 奇数ページでb.pdf  
    "template": [  
        {"*": [  
            {"match": "[0::2]", "src": "./a.pdf"},  
            {"match": "[1::2]", "src": "./b.pdf"}  
        ]}  
    ]  
}
```

コード 3.5: 条件付きテンプレート選択の例

3.3 コンテキスト要素

コンテキスト要素では、PDF テンプレートに配置されたフィールドに設定するデータを宣言します。

フィールドに設定するデータとしては、主にテキスト（テキストフィールドの場合）や画像（ボタンフィールドの場合）などの「値」を指しますが、フォント・表示色・境界線色などの表示属性も同様に指定することができます。Field Reports では、フィールドに設定する値と表示属性を合わせて「フィールド属性」と呼んでいます。

各フィールドは「フィールド名」で一意に特定することができますので、コンテキスト要素では、フィールド名とフィールド属性の組を記述することになります。

3.3.1 単票でのフィールド値の指定

単票では、フィールド名をキーに、フィールド属性を値とした辞書形式のデータを context 要素直下の値として記述します。

フィールド名がテーブル形式の場合は、リストとして記述することもできます。

```
{
  "template": "./mitumori.pdf",

  "context": {
    "date": "平成23年1月22日",
    "number": "10R0001",
    "to": "△△△惣菜株式会社",
    "title": "肉じゃがの材料",
    "delivery_date": "平成23年1月22日",
    "delivery_place": "貴社指定場所",
    "payment_terms": "銀行振込",
    "expiration_date": "発行から3ヶ月以内",
    "stamp1": {"icon": "./stamp.png"},

    "table": [
      ["1", "N001", "牛肉（切り落とし）", "200g", "250円", "500円"],
      ["2", "Y001", "じゃがいも（乱切り）", "3個", "30円", "90円"],
      ["3", "Y002", "にんじん（乱切り）", "1本", "40円", "40円"],
      ["4", "Y003", "たまねぎ（くし切り）", "1個", "50円", "50円"],
      ["5", "Y004", "しらたき", "1袋", "80円", "80円"],
      ["6", "Y005", "いんげん", "1袋", "40円", "40円"]
    ],
    "sub_total": "800円",
    "tax": "40円",
    "total": "840円"
  }
}
```

コード 3.6: 単票でのフィールド値指定の例

3.3.2 複合帳票でのフィールド値の指定

複合帳票の場合は、`template` 要素で宣言した部分フィールド名を利用して、各帳票ごとに分離して記述します。

```
{
  "template": [
    {"header": "./hyousi.pdf"}, {"body": "./mitumori.pdf"}
  ],
  "context": {
    "header": {
      "date": "${NOW}",
      "number": "10R0001",
      "to": "△△△惣菜株式会社",
      "title": "肉じゃがの材料",
      "delivery_date": "2011-03-01",
      "delivery_place": "貴社指定場所",
      "payment_terms": "銀行振込",
      "expiration_date": "発行から3ヶ月以内",
      "total": 840
    },
    "body": {
      "date": "${NOW}",
      "number": "10R0001",
      "to": "△△△惣菜株式会社",
      "title": "肉じゃがの材料",
      "delivery_date": "2011-03-01",
      "delivery_place": "貴社指定場所",
      "payment_terms": "銀行振込",
      "expiration_date": "発行から3ヶ月以内",
      "stamp1": {"icon": "./stamp.png"},
      "table": [
        ["1", "N001", "牛肉（切り落とし）", "200g", 250, 500],
        ["2", "Y001", "じゃがいも（乱切り）", "3個", 30, 90],
        ["3", "Y002", "にんじん（乱切り）", "1本", 40, 40],
        ["4", "Y003", "たまねぎ（くし切り）", "1個", 50, 50],
        ["5", "Y004", "しらたき", "1袋", 80, 80],
        ["6", "Y005", "いんげん", "1袋", 40, 40]
      ],
      "sub_total": 800,
      "tax": 40,
      "total": 840
    }
  }
}
```

コード 3.7: 複合帳票でのフィールド値指定の例

3.3.3 連続帳票でのフィールド値の指定

連続帳票の場合は、フィールド名と属性の組を記述した辞書を並べたリストを context 要素の値として宣言します。

以下の例では、1ページに配置できるテーブルの行数を最大 10 行としているのに対して、データは 14 行分あるので、1 ページには収まらないように見えますが、テーブル分割機能 (4.2) の働きにより自動的に 2 ページに分割されます。

```
{
  "template": [
    {"*": {"src": "./recipe.pdf", "rows": 10}}
  ],
  "context": [
    {
      "title": "ビーフストロガノフ",
      "num": "4",
      "material": [
        ["牛肉", "(バラ、肩ロースなど) 400g"],
        ["玉ねぎ", "1コ"],
        ["マシュルーム", "6コ"],
        ["塩・こしょう", "少々"],
        ["小麦粉", "大さじ2"],
        ["バター", "20g"],
        ["トマトピューレー", "400g(2びん)"],
        ["水", "400cc"],
        ["コンソメ", "顆粒1袋(5g)"],
        ["パプリカ", "小さじ1"],
        ["塩", "小さじ半分"],
        ["さとう", "大さじ2半"],
        ["ブランデー", "大さじ1"],
        ["生クリーム", "少々"]
      ]
    }
  ]
}
```

コード 3.8: 連続帳票でのフィールド値指定の例

3.3.4 フィールド属性の指定

フィールドに値以外の属性を指定する場合は、辞書形式を用います。

```
{
  "resources": {
    "font": {
      "HiraMaruPro-W4": {
        "src": "./fonts/HiraMaruGo_Pro_W4.otf"
      },
      "KouzanBrushFontSousyoOTF": {
        "src": "./fonts/KouzanSoushoOTF.otf"
      }
    }
  },
  "template": {
    "paper": "A4"
  },
  "context": {
    "hello_1": {
      "new": "Tx",
      "value": "こんにちは世界",
      "rect": [100, 700, 400, 750],
      "font": "HiraMaruPro-W4"
    },
    "hello_2": {
      "new": "Tx",
      "value": "こんにちは世界",
      "rect": [100, 600, 400, 650],
      "font": "KouzanBrushFontSousyoOTF"
    }
  }
}
```

コード 3.9: 辞書形式でのフィールド属性指定の例

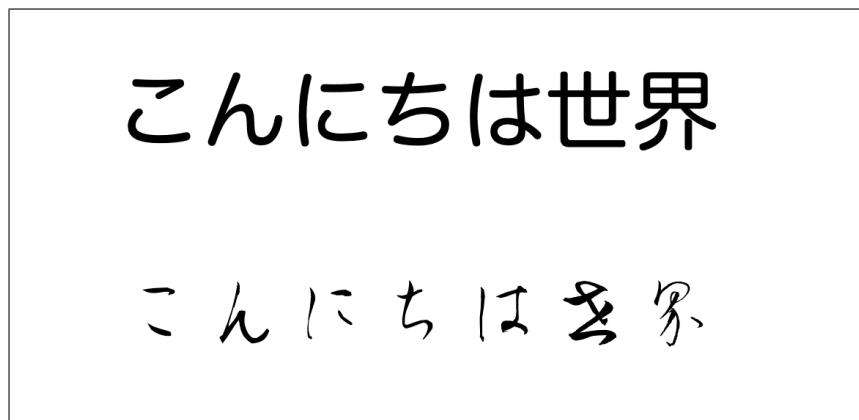


図 3.11: 処理結果

3.4 スタイル要素

`context` 要素ではフィールド名とフィールド属性の組を一対一対応で記述しますが、複数のフィールドの表示属性を一括して指定できると便利な場合もあります。そのような時には、`style` 要素を使用してください。

3.4.1 スタイル指定の例

以下にスタイル指定の例を示します。

```
{
  "template": [
    {"header": "./hyousi.pdf",
     "body.*": {"src": "./mitumori.pdf", "rows": 10}}
  ],
  "context": {
    "header": {
      "date": "${NOW}",
      "delivery_date": "2011-03-01",
      "total": 840
    },
    "body": [
      {
        "date": "${NOW}",
        "delivery_date": "2011-03-01",
        "total": 2699,
        "sub_total": 2570,
        "tax": 128,
        "remark": "...",
        "table": [
          // ...
        ]
      }
    ]
  },
  "style": [
    {"header.date": {"datetime": "GGE年M月D日"}},
    {"header.delivery_date": {"datetime": "GGE年M月D日"}},
    {"header.total": {"format": "###,###円"}},
    {"body.*.date": {"datetime": "GGE年M月D日"}},
    {"body.*.delivery_date": {"datetime": "GGE年M月D日"}},
    {"body.*.total": {"format": "###,###円"}},
    {"body.*.sub_total": {"format": "###,###円"}},
    {"body.*.tax": {"format": "###,###円"}},
    {"body.*.table.*[4:6]": {"format": "###,###円"}},
    {"body.[1:].stamp1": {"visible": false}},
    {"body.[1:].remark": {"visible": false}},
    {"body.*.table.[1::2)": {"background-color": [220, 220, 255]}}
  ]
}
}
```

コード 3.10: スタイル指定の例

3.4.2 セレクタとは

テンプレート上に配置されたフィールドの選択範囲を表現するための記法です。style 要素では、セレクタと field 要素の組としてスタイルを指定します。

完全修飾フィールド名によりフィールドを特定する方法と比較すると、以下の点が異なります。

- フィールド階層構造における共通の祖先を指定することで、その子孫のフィールドを一括して選択できます。
- ワイルドカードを使用することで、パターンマッチングに基づくフィールドの選択ができます。

例えば `style` 指定においては、以下のようなことが可能になります。

- 共通の親を持つフィールドのグループを一括して非表示にする。
- 連続帳票の1ページ目だけに「合計」欄を表示する。2ページ目以降は「合計」欄を空欄にする。
- テーブルの偶数行と奇数行でテキスト表示色を変える。

セレクタの詳細については [5.3](#) を参照してください。

3.5 リソース要素

resource 要素では、フォントと画像のリソースを定義します。

3.5.1 フォントリソース

フォントリソース定義の例

以下にフォントリソース定義の例を示します。

ここで定義したフォント名は、font フィールド属性の値として使用します。

```
{
  "resources": {
    "font": {
      "IPAmjMincho": {
        "src": "./ipamjm.ttf",
        "embed": true,
        "subset": true
      },
      "@KouzanBrushFontSousyoOTF": {
        "src": "./KouzanSoushoOTF.otf",
        "writing-mode": "VerticalRl",
        "embed": true,
        "subset": true
      }
    },
    "template": {
      "paper": "A4"
    },
    "context": {
      "hello_1": {
        "new": "Tx",
        "font": "HiraMaruPro-W4",
        ...
      },
      "hello_2": {
        "new": "Tx",
        "font": "@KouzanBrushFontSousyoOTF",
        ...
      }
    }
}
```

コード 3.11: フォントリソース定義の例

対応フォント形式

Field Reports では、以下のフォント形式に対応しています。

- TrueType（拡張子：*.ttf, *.ttc）
- OpenType（拡張子：*.otf）

フォントの埋め込みについて

フォント埋め込みとはPDFファイルにフォントの字形（グリフ）データを埋め込む機能です。PDFにフォントを埋め込むことで、字形を含むテキスト情報をより確実に伝達できるようになります。

例えば、毛筆体フォントなどのデザインを重視したフォント、OCRフォント・バーコードフォントのように正確な再現が必須となるフォントを使用する場合に特に役立ちます。

フォントを埋め込まない場合、PDFを受け取った相手のコンピュータに同じフォントがインストールされていないと、異なったフォント（代替フォント）で表示されます。適切な代替フォントが見つからない場合、文字位置のずれや文字化け等の問題が発生することがあります。

グリフデータを埋め込むことでPDFのファイルサイズが増えますが、「サブセット化」を有効にすることで、最小限の増加に留めることができます。

フォント埋込機能を有効にするには、`embed` フラグを有効にします。

おは、あけほの。やうやうふくなりゆく「ひぎ
は」、「うしめりて戯だちたる雪のゆくたなびきたる。
友は、ゑ。月のぬはさらなり。夏もなほ。菅の
あくゑびきひたる。また、ただ一つ二つなど、
ほのかにうち光りて行くもをかし。おなど降る
もをかし。

おは、夕季。夕日のせして、山の渓はいとら
うなりたるに、鳥の宿どころへ行くとて、三つ
四つ、二つ三つなど、ゑびきぐせへあはれなり。
まいて在などのをねたるがいとひせく見るは、
いとをかし。日入り果てて、風の音、虫の音な
ど、はたい小べきにあらず。

おは、つとめて。雪の降りたるはい小べきにも
あらず。霜のいとふきも、またさらでも、いと
えきに、ぬなど立さ枝して、寒もて渡るも、い
とづきづきし。空になりて、ぬるくゆるびもて
いけば、火桶の火も、ふき度がちになりて、わ

図 3.12: フォント埋込の例（毛筆体フォント）

3.5.2 画像リソース

画像リソース定義の例

以下に画像リソース定義の例を示します。

ここで定義した画像名称は、image フィールド属性の値として使用します。

```
{
  "resources": {
    "image": {
      "photo1": "./kid0043-009.jpg",
      "photo2": "./kid0054-009.jpg"
    }
  },
  "context": {
    "photo1": {
      "width": 348,
      "height": 230,
      "border-width": 12,
      "border-color": [119, 200, 55],
      "border-join-style": "BevelJoin",
      "rotation": 12.69,
      "image": "photo1"
    },
    "photo2": {
      "width": 330,
      "height": 225,
      "border-width": 12,
      "border-color": [255, 124, 128],
      "border-join-style": "RoundJoin",
      "rotation": -5.74,
      "image": "photo2"
    }
  },
  ...
}
```

コード 3.12: 画像リソース定義の例

対応画像形式

Field Reports では、以下の画像形式に対応しています。

- PNG
- BMP
- JPEG
- JPEG2000
- PDF

透過データ（ α チャンネル）を持った PNG, BMP 形式に対応しています。

PNG 形式には以下の制限があります。

- インターレースモード形式には未対応です。
- α チャンネルを持ち、かつビット深度 4 以下の形式（16 色以下のグレースケール・インデックスカラーなど）には未対応です。

3.5.3 リソース URL 指定について

画像・PDF テンプレートの取得先として、任意の URL を指定することが可能です。

- ローカルファイル
- data URI scheme
- URL

3.6 可変テーブル

コンテンツの量に合わせて、行数を可変にできるテーブル形式です。

連続帳票形式とすることで、複数ページにまたがるテーブルも作成できます。

3.6.1 可変テーブルの構成

可変テーブルは、ヘッダー部、ボディー部、フッターボードで構成されます（図 3.13）。



図 3.13: 横組み可変テーブルの構成

テーブル全体の組方向は、フィールドに指定されたフォントの組み方向（writing-mode）により決定されます。横組みフォントを指定すれば横組みテーブルに、縦組みフォントを指定すれば縦組みテーブルとなります。

縦組みテーブルでは、右から左方向に行番号が、上から下方向に列番号が繰り上がります（図 3.14）。

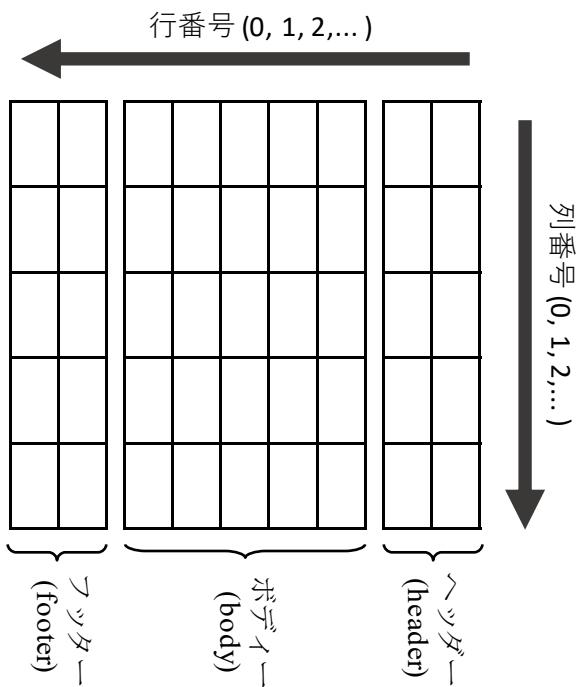


図 3.14: 縦組み可変テーブルの構成

可変テーブルを構成する個々のコマ要素を「セル」と呼び、以下の形式で「セル名」が自動的に付けられます。

- | | |
|--------------|------------------------------------|
| ヘッダー部： | テーブル名 . <i>header</i> . 行番号 . 列番号 |
| ボディ一部（相対形式）： | テーブル名 . <i>body</i> . 行番号 . 列番号 |
| ボディ一部（絶対形式）： | テーブル名 . \$ <i>body</i> . 行番号 . 列番号 |
| フッター部： | テーブル名 . <i>footer</i> . 行番号 . 列番号 |

ボディ一部のセル名には、相対形式と絶対形式があります。

絶対形式は、連続帳票で分割されたテーブルに対し、通し番号で行を指定する場合に使用します（図 3.15）。

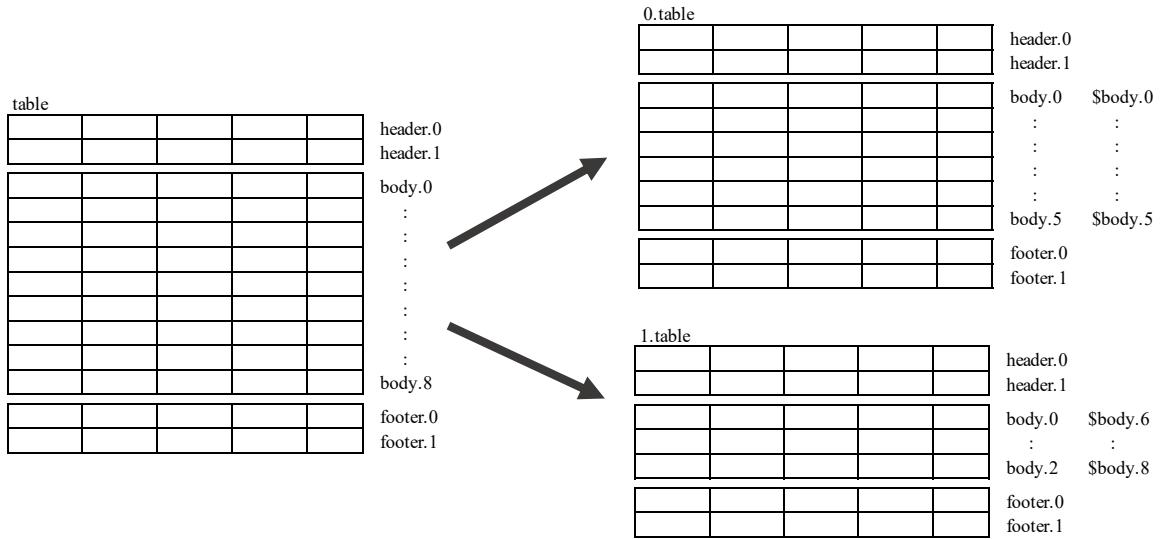


図 3.15: 連続帳票でのセル名

セル名は、スタイルを指定する際に、セレクタとして使用できます。

3.6.2 セル値の指定

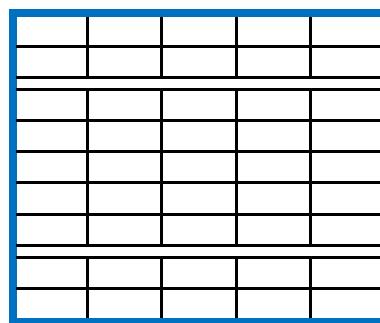
テーブルを構成する各セルの値として、ヘッダー部・ボディ一部・フッター部に分けて、セル値を2次元配列（行列）形式で配置します。

```
{
  "context": {
    "table": {
      ...
      "header": [
        "商品コード", "品名", "数量", "単価", "金額"
      ],
      "body": [
        ["1002", "小松菜", "2", 10000, 200000],
        ["1003", "トマト", "3", 10000, 400000],
        ["1004", "すいか", "4", 10000, 400000],
        ...
      ],
      "footer": [
        ["", "", "", "小計", 999999],
        ["", "", "", "消費税等", 999899],
        ["", "", "", "合計", 999999]
      ]
    }
  },
  ...
}
```

コード 3.13: セル値指定の例

3.6.3 境界線の指定

テーブル全体を囲む境界線は、フィールドの境界線スタイル属性（[5.9.4](#)）で指定します（[図 3.16](#)）。



A 10x5 grid of cells, representing a table with 10 columns and 5 rows. The entire grid is enclosed within a single, thick blue border.

図 3.16: テーブル境界線

行間・列間境界線は、border-columns, border-rows でそれぞれ指定します（[図 3.17](#)）。

The figure consists of two separate tables. The left table has 5 rows and 5 columns, with thick blue horizontal lines separating each row. The right table also has 5 rows and 5 columns, but it features thick blue vertical lines separating each column.

図 3.17: 行間境界線と列間境界線

ヘッダー部とボディー部の間、ボディー部とヘッダー部の間の境界線は、`border-header`, `border-footer` で指定します（図 3.18）。

The diagram shows a table with 5 rows and 5 columns. A thick blue horizontal line at the very top is labeled "border-header". Another thick blue horizontal line at the very bottom is labeled "border-footer".

図 3.18: ヘッダー境界線とフッター境界線

3.6.4 フィールド属性の指定

テキストフィールドと同様に、セル値としてテキスト値以外のフィールド属性を指定することができます。可変テーブルに対して適用可能なフィールド属性については、表 5.23 を参照してください。

すべてのセルに対する指定方法

すべてのセルに共通するフィールド属性を指定する場合は、可変テーブルの `field` 要素として記述します。

```
{
  "context": {
    "table": {
      "new": "Tbl",
      "font-size": 10,
      "rect": [55, 283, 539, 661],
      "multiline": true,
      "column-widths": ["20%", "30%", "10%", "20%", "20%"],
      "row-height": 25,
      "cell-padding": 4,
      "border": {"width": 2, "color": "Blue"},
      "border-rows": {"width": 0.5, "color": "Silver"},
      "border-columns": {"width": 0.5, "style": "Dashed", "dash": [2]},
      "border-header": {"width": 1, "color": "Blue"},
      "border-footer": {"width": 1, "color": "Blue"},
      ...
    }
  },
  ...
}
```

コード 3.14: すべてのセルに対する指定の例

個別のセルに対する指定方法

特定のセルに個別のフィールド属性を指定する場合は、field 要素を辞書形式で与えます。

```
{
  "context": {
    "table": {
      "header": [
        [
          "商品コード",
          {"value": "品名", "color": "blue"},
          {"value": "数量", "font-stretch": 1.5},
          "単価",
          "金額"
        ]
      ],
      ...
    }
  },
  ...
}
```

コード 3.15: 個別のセルに対する指定の例

スタイルによる一括指定

複数のセルに対し、行単位・列単位等でフィールド属性を指定する場合は、スタイルを利用します。

```
{  
    ...  
    "style": [  
        {"table.header.*.*": {"text-align": "Center", "vertical-align": "Middle"}},  
        {"table.body.*.*": {"vertical-align": "Middle"}},  
        {"table.body.*.[2:]" : {"text-align": "Right", "format": "#,#"}},  
        {"table.footer.*.*": {"vertical-align": "Middle"}},  
        {"table.footer.*.[3]": {"multiline": false, "text-align": "Justify"}},  
        {"table.footer.*.[4]": {"text-align": "Right", "format": "#,#"}}  
    ]  
}
```

コード 3.16: スタイルによる一括指定の例

3.6.5 記述例

可変テーブルの記述例を以降に示します。

可変テーブルに対応するフィールドが PDF に仕様に存在しないため、テンプレートに配置されたフィールドを可変テーブルのプレースホルダとして利用することができません。可変テーブルでは、座標を含めたすべてのテーブル構成要素をレンダリングパラメータで記述します。

単票

単票における可変テーブルの記述例を以下に示します。

```
{
  "template": {"paper": "A4"},

  "context": {
    "table": {
      "new": "Tbl",
      "font-size": 10,
      "rect": [55, 283, 539, 661],
      "multiline": true,
      "column-widths": ["20%", "30%", "10%", "20%", "20%"],
      "row-height": 25,
      "cell-padding": 4,
      "border": {"width": 2, "color": "Blue"},

      "border-rows": {"width": 0.5, "color": "Silver"},

      "border-columns": {"width": 0.5, "style": "Dashed", "dash": [2]},

      "border-header": {"width": 1, "color": "Blue"},

      "border-footer": {"width": 1, "color": "Blue"},

      "header": [
        ["商品コード", "品名", "数量", "単価", "金額"]
      ],
      "body": [
        ["1002", "小松菜", "2", 10000, 20000],
        ["1003", "トマト", "3", 10000, 30000],
        ["1004", "すいか", "4", 10000, 40000],
        ...
      ],
      "footer": [
        ["", "", "", "小計", 999999],
        ["", "", "", "消費税等", 999899],
        ["", "", "", "合計", 999999]
      ]
    }
  },
  "style": [
    {"table.header.*.*": {"text-align": "Center", "vertical-align": "Middle"}},
    {"table.body.*.*": {"vertical-align": "Middle"}},
    {"table.body.*.[2:]" : {"text-align": "Right", "format": "#,#"}},
    {"table.footer.*.*": {"vertical-align": "Middle"}},
    {"table.footer.*.[3]": {"multiline": false, "text-align": "Justify"}},
    {"table.footer.*.[4]": {"text-align": "Right", "format": "#,#"}}
  ]
}
}
```

コード 3.17: 可変テーブル記述例（単票）

単票可変テーブルでは、rect 属性で指定した矩形領域に行が収まらない場合、内容物が見切れます。

商品コード	品名	数量	単価	金額
1002	小松菜	2	10,000	20,000
1003	トマト	3	10,000	30,000
1004	すいか	4	10,000	40,000
1005	キャベツ	5	10,000	50,000
1006	レタス	1	10,000	10,000
1007	なす	2	10,000	20,000
1008	きゅうり	3	10,000	30,000
1009	大葉	4	10,000	40,000
1010	パジル	5	10,000	50,000
1011	パセリ	1	10,000	10,000
小計				999,999
消費税等				999,899
合計				999,999

図 3.19: 可変テーブル出力例（単票）

連続帳票

テンプレート要素とコンテキスト要素を連続帳票形式で記述することで、複数ページにまたがる可変テーブルとすることができます。

```
{
  "template": [
    {"*": {"paper": "A4"}}
  ],
  "context": [
    {
      "table": {
        "new": "Tbl",
        "font-size": 10,
        "rect": [55, 283, 539, 661],
        "multiline": true,
        "column-widths": ["20%", "30%", "10%", "20%", "20%"],
        "row-height": 25,
        "cell-padding": 4,
        "border": {"width": 2, "color": "Blue"},
        "border-rows": {"width": 0.5, "color": "Silver"},
        "border-columns": {"width": 0.5, "style": "Dashed", "dash": [2]},
        "border-header": {"width": 1, "color": "Blue"},
        "border-footer": {"width": 1, "color": "Blue"},
        "header": [
          ["商品コード", "品名", "数量", "単価", "金額"]
        ],
        "body": [
          ["1002", "小松菜", "2", 10000, 200000],
          ["1003", "トマト", "3", 10000, 400000],
          ["1004", "すいか", "4", 10000, 400000],
          ...
          ["1002", "大根", "1", 9000, 9000],
          ["1003", "トマト", "3", 500, 1500],
          ["1004", "すいか", "4", 7000, 28000]
        ],
        "footer": [
          ["", "", "", "小計", 999999],
          ["", "", "", "消費税等", 999899],
          ["", "", "", "合計", 999999]
        ]
      }
    }
  ],
  "style": [
    {"*.table.header.*.*": {"text-align": "Center", "vertical-align": "Middle"}},
    {"*.table.body.*.*": {"vertical-align": "Middle"}},
    {"*.table.body.*.[2:]": {"text-align": "Right", "format": "#,#"}},
    {"*.table.footer.*.*": {"vertical-align": "Middle"}},
    {"*.table.footer.*.[3]": {"multiline": false, "text-align": "Justify"}},
    {"*.table.footer.*.[4]": {"text-align": "Right", "format": "#,#"}}
  ]
}
}
```

コード 3.18: 可変テーブル記述例（連続帳票）

rect 属性で指定した矩形領域に行が収まらない場合に、テーブルをページ分割します。header, footer 行は、分割後の各ページで繰り返されます（図 3.20）。

商品コード	品名	数量	単価	金額
1002	小松菜	2	10,000	20,000
1003	トマト	3	10,000	30,000
1004	すいか	4	10,000	40,000
1005	キャベツ	5	10,000	50,000
1006	レタス	1	10,000	10,000
1007	なす	2	10,000	20,000
1008	きゅうり	3	10,000	30,000
1009	大葉	4	10,000	40,000
1010	バジル	5	10,000	50,000
1011	バセリ	1	10,000	10,000
		小計		999,999
		消費税等		999,899
		合計		999,999

商品コード	品名	数量	単価	金額
1002	大根	1	9,000	9,000
1003	トマト	3	500	1,500
1004	すいか	4	7,000	28,000
		小計		999,999
		消費税等		999,899
		合計		999,999

図 3.20: 可変テーブル出力例（連続帳票）

3.7 リッチテキスト（Professional 版）

Acrobat Pro では、テキストフィールドのプロパティで「リッチテキスト形式の許可」（図 3.21）にチェックを入れると、フィールドに入力したテキストのスタイルを指定することができるようになります。これは、PDF 1.5 より追加された “Rich Text Strings” の仕様に基づく機能です。



図 3.21: 「リッチテキストの形式を許可」プロパティ

Field Reports では、“Rich Text Strings” をベースとして独自の XML 文書を定義しています。「リッチテキスト」を利用する事で、きめ細かくスタイルを指定する事が可能になります。

リッチテキストを有効にするには、テキスト・バリュー属性（5.9.5）において、richtext 属性を有効とし、value 属性の値として XML 形式のリッチテキストを記述します。

以下に、リッチテキストを利用したレンダリングパラメーター（JSON 形式）の記述例を示します。

```
{
  "resources": {
    "font": {
      "HiraMinPro-W6": {
        "src": "./fonts/HiraMin_Pro_W6.otf",
        "embed": true,
        "subset": true
      }
    }
  },
  "template": {
    "paper": "A4"
  },
  "context": {
    "text": {
      "new": "Tx",
      "richtext": true,
      "font": "HiraMinPro-W6",
      "font-size": 12,
      "value": "<body><p>春は、<span style='font-size:2em;color:cmyk(0,0.5,0.5,0);'>あけぼの</span>。やうやう白くなりゆく</span>少し明りて紫だちたる雲の細くたなびきたる。</p><p><ruby>夏は、夜<rt>なつはよる</rt></ruby>。月の頃はさらなり。闇もなほ。螢の多く飛び違ひたる。また、ただ一つ二つなど、ほのかにうち光りて行くもをかし。雨など降るもをかし。</p></body>",
      "multiline": true,
      "text-align": "Justify",
      "vertical-align": "Top",
      "line-height": 1.2,
      "padding": 4,
      "rect": [100, 680, 400, 800]
    }
  }
}
```

コード 3.19: リッチテキスト記述例

出力結果を図 3.22 に示します。

春は、**あけぼの**。やうやう白くなりゆく山ぎは少し明りて紫だちたる雲の細くたなびきたる。
なつはよる 夏は、夜。月の頃はさらなり。闇もなほ。螢の多く飛び違ひたる。また、ただ一つ二つなど、ほのかにうち光りて行くもをかし。雨など降るもをかし。

図 3.22: リッチテキストの出力例

リッチテキストの詳細については、第 6 章を参照してください。

3.8 拡張漢字の利用

3.8.1 追加面に格納された Unicode 文字の指定

日本語入力ソフトウェアで変換できない拡張漢字は、 Unicode コードポイントを用いて指定することができます。

エスケープシーケンスによる指定

プログラミング言語で用意されている Unicode エスケープシーケンスを利用する方法です。

Python, C#では、 \uhhhh または \Uhhhhhhhh で 16 ビットまたは 32 ビットの Unicode コードポイントが指定できます。

Java では、 \uhhhh がありますが、 16 ビット Unicode コードポイントしか指定できません。追加面の文字を指定する場合は、 2 文字のサロゲートペアに分解して指定してください。

Ruby, PHP では、 Unicode 用のエスケープシーケンスは特に用意されていないので、 \xhh などのバイト文字用のエスケープシーケンスを利用して、 UTF-8 バイト列を埋め込んでください。

JSON 形式でレンダリングパラメータを記述する場合は、 16 ビット形式の \uhhhh が利用できます。Field Reports では、 JSON の独自拡張として、 \Uhhhhhhhh 形式の 32 ビット Unicode コードポイントも利用することができます。

文字参照による指定

文字参照により Unicode コードポイントを指定することもできます。 &#dddd; または &#xhhhhhh; という書式により、 10 進数または 16 進数による指定を行うことができます。

文字参照はフィールド単位で有効・無効を切り替えます。デフォルトでは無効になっているので、 利用する場合は明示的に有効にする必要があります。

使用例

以下にエスケープシーケンスと文字参照を利用して、 サロゲートペア（4 バイト文字）で表現される漢字を表示する例を示します。

```
{
  "resources": {
    "font": {
      "IPAmjMincho": {
        "src": "../fonts/ipamjm.ttf",
        "embed": true,
        "subset": true
      }
    }
  },
  "template": {"size": "A4"},

  "context": {
    "text": [
      {
        "new": "Tx",
        "font": "IPAmjMincho",
        "font-size": 24,
        "multiline": true,
        "rect": [50, 650, 550, 750],
        "charref": true,
        "value": "\U0002000B\U00020089\U000200A2\U000200A4&x201A2;&x20213;&x2032B;
        &x20371;&x20381;&x203F9;&x2044A;&x20509;&x205D6;&x20628;&x2074F;
        &x20807;&x2083A;&x208B9;&x2097C;&x2099D;\n
        &x2A716;&x2A729;&x2a72a;&x2a72c;&x2a738;&x2a73d;&x2a746;&x2a752;
        &x2a758;&x2a75f;\n
        &x2B740;&x2B741;&x2B742;&x2B743;&x2B744;&x2B745;&x2B746;&x2B747;
        &x2B749;&x2B74A;&x2B74C;&x2B74D;"}
    ]
  }
}
```

コード 3.20: サロゲートペアの使用例

丈々辰自ヘ倅俾集俾僵儼矣浴几劔劉勅勳斗卓
 韻勾勾余併金傍假隨傍
 五尹所爾久亥今余倉傳獎岡

図 3.23: サロゲートペアの使用例

3.8.2 異体字セレクタによる異体字の指定

異体字セレクタとは

Unicode の漢字統合の原則により、複数の自体のバリエーションを持つ文字であってもひとつのコードポイントに統合されるのが基本です。例えば多くの字体が存在することで有名な渡邊の「邊」で、実際にコードポイントが割り当てられてるのは、邊 (U+9089), 邊 (U+908A) の 2 文字だけです。

人名・地名などを扱うためには字体のバリエーションを実際に区別する必要がありますが、漢字統合の原則によりむやみに増やすことができません。そこで、漢字統合の原則を守りつつ字体を区別する方法として考えだされたのが異体字セレクタと呼ばれる特殊な文字コードです。

日本語の場合、U+E0100～U+E01EF の範囲のコードを異体字セレクタとして使用します。基本となる正体字の文字コードに続けて異体字セレクタを配置することで、何番目の異体字かを表現します。

使用例

以下に、IPAmj 明朝フォントを使って邊の異体字を列挙した例を示します。U+E0100～U+E01EF は 2 バイトで表現できないので、独自形式の JSON で記述しています。

```
{
  "resources": {
    "font": {
      "IPAmjMincho": {
        "path": "./ipamjm.ttf",
        "embed": true,
        "subset": true
      }
    }
  },
  "template": {"paper": "A4"},

  "context": {
    "text": [
      {
        "new": "Tx",
        "font": "IPAmjMincho",
        "multiline": true,
        "rect": [50, 500, 550, 630],
        "value": "\u0009\u000E010F\u0009\u000E0119\u0009\u000E011B\u0009\u000E011A\u0009
        \u0009\u000E011C\u0009\u000E011D\u0009\u000E0117\u0009\u000E0116\u0009\u000E0115\u0009\u000E
        \u0009\u000E0114\u0009\u000E0118\u0009\u000E0113\u0009\u000E0112\u0009\u000E0111\u0009\u000E0110\n\u0009
        \u0009\u000E0108\u0009\u000E0109\u0009\u000E010A\u0009\u000E010B\u0009\u000E010C\u0009\u000E
        \u0009\u000E010D\u0009\u000E010E\u0009\u000E010F\u0009\u000E0110\u0009\u000E0110\n葛飾区 葛\u0009\u000E0102城市／蓮田市 蓼
        \u0009\u000E0104田市"
      }
    ]
  }
}
```

コード 3.21: 異体字セレクタの使用例

邊邊邊邊邊邊邊邊邊邊邊邊
 邊邊邊邊邊邊邊邊邊邊
 葛飾区 葛城市／蓮田市 蓼田市

図 3.24: 異体字セレクタの使用例

3.8.3 グリフ直接指定

CID によるグリフ指定

使用するフォントが OpenType の CJK フォント（拡張子：*.otf）であれば、CID をキーとしてグリフを指定することができます。CID とは、CID フォントが内蔵するすべてのグリフを一意に識別するために、Adobe 社が付与した番号です。その番号体系は、日本語 CID フォントであれば Adobe-Japan1 文字コレクションに

もとづいています。

CID によるグリフ指定は、Field Reports 独自の以下の実体参照形式により行います。

&#dddd; または &#xhhhhh;

グリフ名によるグリフ指定

使用するフォントで「グリフ名」が定義されていれば、グリフ名によるグリフ指定を行うこともできます。Field Reports 独自で定義した実体参照形式によりグリフを指定することができます。

&@<グリフ名>;

グリフ名が定義されているかどうかは、ttfdump などのツールで、TrueType フォントの「post」テーブルの内容をダンプすると確認することができます。

使用例

以下に、IPAmj 明朝フォントを使ってグリフ名によるグリフ指定を行った例を示します。

```

{
  "resources": {
    "font": {
      "IPAmjMincho": {
        "src": "./ipamjm.ttf",
        "embed": true,
        "subset": true
      }
    }
  },
  "template": {"size": "A4"},

  "context": {
    "text": [
      {
        "new": "Tx",
        "font": "IPAmjMincho",
        "font-size": 24,
        "multiline": true,
        "rect": [50, 350, 550, 450],
        "charref": true,
        "value": "&@mj000007;&@mj000008;&@mj000012;&@mj000022;&@mj000023;&@mj000028;&@mj000029;&@mj000036;&@mj000037;&@mj000045;&@mj000046;&@mj000047;&@mj000048;&@mj000073;&@mj000074;&@mj000089;&@mj000105;&@mj000106;&@mj000129;&@mj000130;&@mj000143;&@mj000144;&@mj000145;&@mj000146;&@mj000156;&@mj000157;&@mj000175;&@mj000176;&@mj000183;&@mj000184;&@mj000185;&@mj000206;&@mj000207;&@mj000208;&@mj000209;&@mj000241;&@mj000242;&@mj000264;&@mj000265;&@mj000266;&@mj000267;&@mj000269;&@mj000270;&@mj000276;&@mj000277;&@mj000278;&@mj000302;&@mj000303;&@mj000309;&@mj000310;&@mj000311;&@mj000312;&@mj000317;&@mj000318;&@mj000319;&@mj000322;&@mj000333;&@mj000380;&@mj000381;&@mj000405;&@mj000406;"}
    ]
  }
}

```

コード 3.22: グリフ名指定の使用例

図 3.25: グリフ名指定の使用例

第4章

帳票生成

4.1 帳票生成処理の概要

図 4.1 は、Field Reports がレンダリングパラメータと PDF テンプレートを元に PDF 帳票を作成するまでの処理の流れを示したものです。

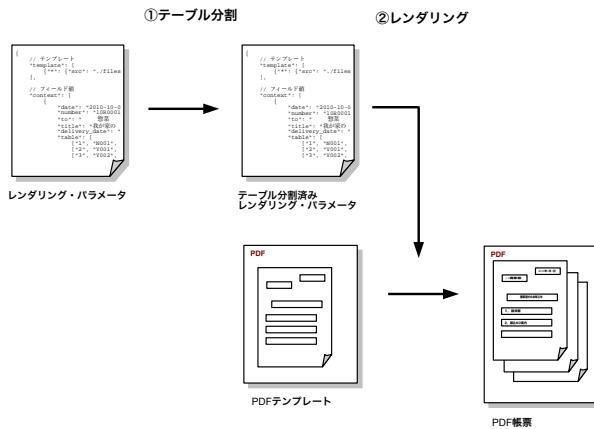


図 4.1: 処理の流れ (全体)

4.1.1 テーブル分割処理

最初に前処理として、レンダリングパラメータにテーブル分割処理 (4.2) を施します。具体的には、レンダリングパラメータの中から可変テーブルまたはテーブル形式のデータを探し出し、1ページに収まる分量のテーブルに分割します。

ここまで処理で、連続帳票部分に必要なページ数が確定します。

4.1.2 レンダリング処理

次に、PDF テンプレートとテーブル分割処理済みのレンダリングパラメータを合成して、ひとつの PDF 帳票としてまとめます。

図 4.2 は、このレンダリング処理部分を詳細に図示したものです。

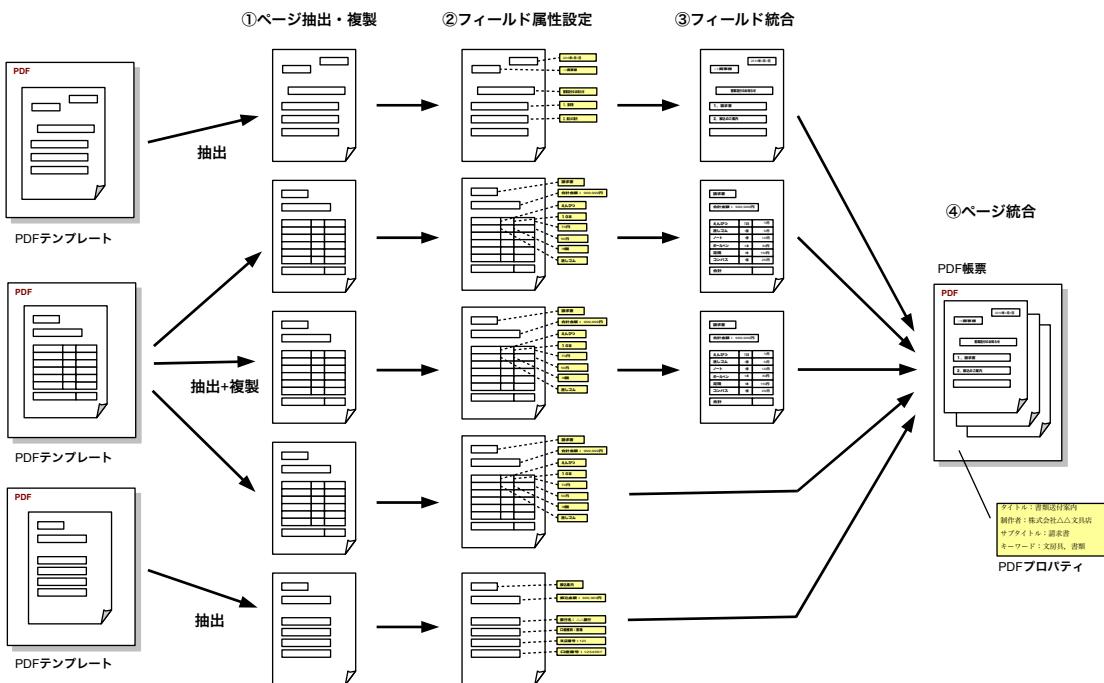


図 4.2: 処理の流れ（レンダリング部分）

ページ抽出・複製処理

レンダリングパラメータの定義にしたがって、PDF テンプレートから必要なページを抜き出します。連続帳票の場合は、さらに抽出したページをテーブル分割処理で確定したページ数分だけ複製します。

PDF テンプレートにあらかじめ配置されているフィールドは、この段階ではページ抽出・複製後もそのまま複製されます（ただし、複合帳票・連続帳票の場合は、フィールド名の名前空間の階層が深くなります）。

フィールドを動的に配置する場合は、この段階で新規にフィールドを生成します。

フィールド属性設定処理

各ページに配置されたフィールドに、テキスト・表示色・境界線などのフィールド属性を設定します。

フィールド属性は、コンテキスト要素もしくはスタイル要素としてレンダリングパラメータに記述されています。

フィールド属性には、PDF の仕様で規定された「基本フィールド属性」と、Field Reports が独自に拡張した「拡張フィールド属性」があります。

フィールド統合処理

フィールドに設定されたフィールド属性値を元にして、PDF 描画命令列（外観ストリーム）を生成します。

フィールド統合が有効な場合には、生成された外観ストリームはページのコンテンツ・ストリーム（テンプ

レートの下絵)と統合され、元のフィールドは削除されます。この処理によりフィールドの外観は恒久化され、基本的に変更ができないなります。

フィールド統合が無効な場合はフィールドが残りますが、フィールドの新しい外観として、ここで生成された外観ストリームが設定されます。

フィールドに関連付けられた外観ストリームは、PDF ビューア (Acobe Reader 等) で開いた時の初期の外観として利用されますが、PDF ビューア依存の任意のタイミングで再構築されます。フィールド統合を行わない場合は、拡張フィールド属性を使用せずに基本フィールド属性の範囲内で使用してください。

ページ統合処理

各ページを統合して、ひとつの PDF 帳票にまとめます。

結合した PDF に対して、プロパティを設定することができます (表 4.1)。Adobe Acrobat の「プロパティ」に概ね対応するプロパティ項目を設定することができますが、実際に設定可能な項目については [5.10](#) を参照してください。

表 4.1: 設定可能なプロパティの一覧

種類	設定可能な値
文書情報	タイトル、作成者、サブタイトル、キーワード、アプリケーション、PDF 変換、作成日・更新日、XMP 形式メタデータ
セキュリティ	パスワードによるアクセス制限 暗号アルゴリズム: RC4(40 ビット/128 ビット), AES(128 ビット)
開き方	レイアウトと倍率、ウインドウオプション、ユーザー・インターフェースオプション、表示領域、印刷領域、印刷ダイアログプリセット
その他	Flate 圧縮、WEB 表示用に最適化

4.2 テーブル分割処理の詳細

レンダリングパラメータの中からテーブル形式のデータを探し出し、データの行数をカウントします。1ページに収まらないようであればテーブルを分割します。1ページ中の最大行数は、レンダリングパラメータの中に定数として記述されます。

以降に、1ページ10行でテーブル分割した場合の実行例を示します。テーブルの分割に連動して、他のフィールドも複製されることに注目してください。

一方、改ページの位置を手動で制御したい場合には、最初から分割後の様に複数ページに分けたデータを作成することで、自動テーブル分割処理を抑制することができます。

```
{
  "template": {"*": {"src": "./recipe.pdf", "rows": 10}},
  "context": [
    {
      "title": "ビーフストロガノフ",
      "num": "4",
      "material": [
        ["牛肉", "(バラ、肩ロースなど) 400g"],
        ["玉ねぎ", "1コ"],
        ["マッシュルーム", "6コ"],
        ["塩・こしょう", "少々"],
        ["小麦粉", "大さじ2"],
        ["バター", "20g"],
        ["トマトピューレー", "400g (2びん)"],
        ["水", "400cc"],
        ["コンソメ", "顆粒1袋(5g)"],
        ["パプリカ", "小さじ1"],
        ["塩", "小さじ半分"],
        ["さとう", "大さじ2半"],
        ["ブランデー", "大さじ1"],
        ["生クリーム", "少々"]
      ]
    }
  ]
}
```

コード 4.1: テーブル分割前のレンダリングパラメータ

```
{
  "template": {"*": {"src": "./recipe.pdf", "rows": 10}},
  "context": [
    {
      "title": "ビーフストロガノフ",
      "num": "4",
      "material": [
        ["牛肉", "(バラ、肩ロースなど) 400g"],
        ["玉ねぎ", "1コ"],
        ["マッシュルーム", "6コ"],
        ["塩・こしょう", "少々"],
        ["小麦粉", "大さじ2"],
        ["バター", "20g"],
        ["トマトピューレー", "400g (2びん)"],
        ["水", "400cc"],
        ["コンソメ", "顆粒1袋 (5g)"],
        ["パプリカ", "小さじ1"]
      ],
      "title": "ビーフストロガノフ",
      "num": "4",
      "material": [
        ["塩", "小さじ半分"],
        ["さとう", "大さじ2半"],
        ["ブランデー", "大さじ1"],
        ["生クリーム", "少々"]
      ]
    }
  ]
}
```

コード 4.2: テーブル分割後のレンダリングパラメータ

4.3 レンダリング処理の詳細

4.3.1 テキストフィールドの外観生成

以下のフィールド属性を主なパラメータとして、テキストの外観を生成します。

- 値（テキスト）
- フォント
- フォントサイズ
- テキストの色
- 整列方法
- 行間隔
- 書式指定
- 複数行指定
- 座標（左下、右上）

フォント

システム定義フォント（[5.9.6](#)）またはリソースとして定義されたフォントが指定できます。

フォントサイズ

フォントサイズが自動（0pt）の場合には、テキストがフィールドの矩形に収まる最大のフォントサイズを自動計算します。ただし複数行指定が有効な場合は、10.5 ポイントに固定とします。

整列方法

左寄せ・中央寄せ・右寄せに加えて、均等割付の指定が可能です。



図 4.3: 横組みでのテキストの整列

縦組みでは、上寄せ・中央寄せ・下寄せ・均等割付の指定ができます。

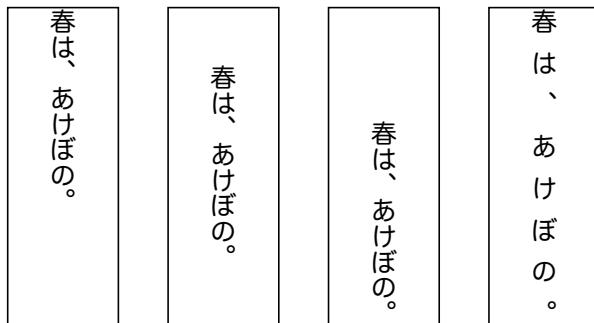


図 4.4: 縦組みでのテキストの整列

組版処理

JIS X 4051 にもとづいた、行分割・禁則処理・割付等の日本語組版を行います。欧文テキストに対しては、ハイフネーションを考慮した行分割を行います。

さらに Professional 版では、リッチテキスト (6) を用いて、文字単位の装飾を施すことができます。

4.3.2 ボタン（画像）フィールドの外観生成

以下のフィールド属性をパラメータとして、ボタン（画像）フィールドの外観を生成します。

- アイコン（画像）
- 座標（左下、右上）

画像の縦横比を保存した上で、ボタン（画像）フィールドの中央に画像を配置します。画像のサイズは、フィールドの矩形に収まる最大のサイズとします。「レイアウト」「アイコンの配置」等の属性が設定してあっても無視します。

また、透過情報（ α チャンネル）を持った PNG/BMP 画像に対応しています。



図 4.5: 透過画像の利用例

4.3.3 境界線の外観生成

以下のフィールド属性をパラメータとして、境界線と背景の塗りつぶしの外観を生成します。

- 境界線の幅
- 境界線の色
- 塗りつぶしの色
- 塗りつぶしのスタイル

図 4.6 に、境界線と背景色の描画例を示します。

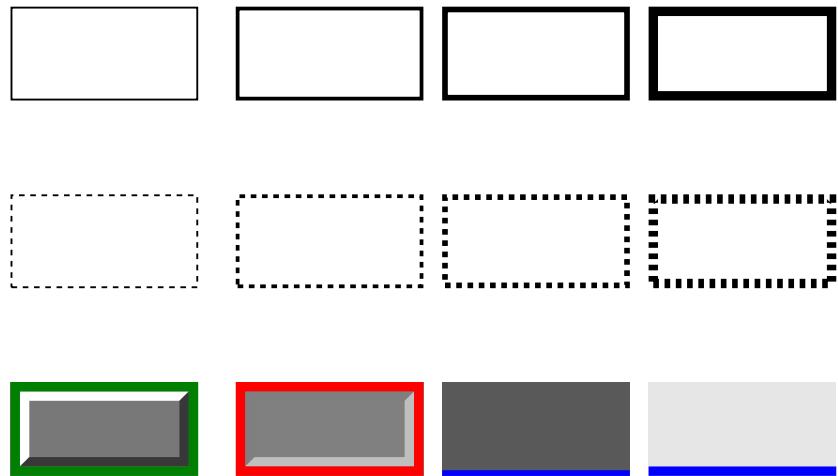


図 4.6: 境界線と背景色

4.3.4 回転角度

フィールドに任意の回転角度を設定することができます。

基本フィールド属性の「向き」を指定した場合、フィールドの内容物の描画方向が 90 度単位で回転します。

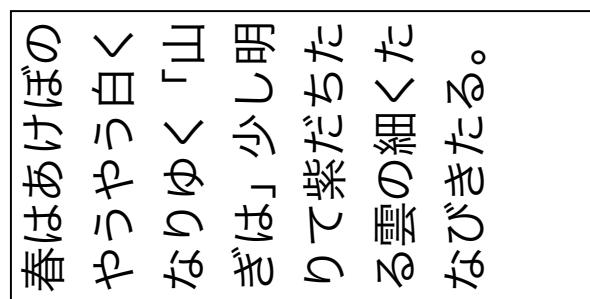


図 4.7: 基本フィールド属性の「向き」を 90 度に指定

4.3.5 座標変換

拡張フィールド属性の「回転角度」を指定した場合、フィールド自体が任意の角度で回転します。



図 4.8: 拡張フィールド属性の「回転角度」を指定

さらに座標変換行列を直接指定すれば、任意のアフィン変換を掛けることができます。

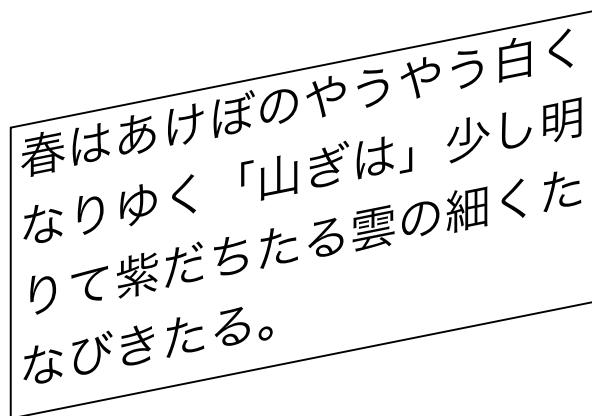


図 4.9: 拡張フィールド属性の「座標変換行列」を指定

4.3.6 透明度

透明度の設定により、フィールドを重ねて表示することができます。すかし・印影などの表現に利用できます。

春はあけぼのやうやうやう白
夏は夜。月の頃はさ
なりゆく「闇きは」少し明
らなり。闇もなほ。
りで紫だちが違ひた
の多く飛びたる雲の細くた
なびきたる。

図 4.10: 透明度を指定しての重ねあわせ表示

4.3.7 ブレンドモード

フィールドを重畠表示する際に、下記のブレンドモードを指定した合成ができます。

- 通常
- 乗算
- スクリーン
- オーバーレイ
- ソフトライト
- ハードライト
- 覆い焼きカラー
- 焼き込みカラー
- 比較（暗）
- 比較（明）
- 差の絶対値
- 除外
- 色相
- 彩度
- カラー
- 輝度



図 4.11: ブレンドモードを指定して背景画とテキストを合成

第 5 章

レンダリングパラメータ

5.1 基本データ

表 5.1 に示す 基本データを組み合わせてレンダリングパラメータを記述します。

表 5.1: 基本データ

基本データ型	例
数値	42, 3.14
真理値	True, False
文字列	“文字列”
列挙値	Left, CropBox
辞書	{"title": “請求書”, “合計”: 10000}
リスト	[1, 2, 3]

辞書の要素は、取り出す際に並び順が維持されないものとします。

5.1.1 JSON で記述する場合

JSON でレンダリングパラメータを記述する際には、基本データ型と JSON のデータ型を表 5.2 のように対応付けます。

表 5.2: JSON データ型との対応

基本データ型	JSON データ型
数値	整数または浮動小数点数
真理値	真理値
文字列	文字列
列挙値	文字列
辞書	オブジェクト
リスト	配列

整数は、10 進記法に限ります。8 進・16 進記法は使用できません。浮動小数点数としては、1.0e-10 のような指数表記も可能です。

真理値としては、true と false が使用できます。

文字列は、ダブルコーテーションでくくります。文字のエンコーディングは、UTF-8 とします。表 5.3 のエ

スケープ文字を含めることができます。

辞書は、オブジェクトに対応付けます。オブジェクトは、キーと値のペアをコロンで対にして、これらの対をコンマで区切ってゼロ個以上列挙し、全体を中カッコでくくることで表現します。キーとして使うデータの型は文字列に限ります。

リストは、配列に対応付けます。配列はゼロ個以上の値をコンマで区切って、角カッコくくることで表現します。

表 5.3: JSON で利用可能なエスケープ文字

エスケープ文字	意味
\\"	バックスラッシュ (\)
\\"	二重引用符 ("")
\\\	スラッシュ (/)
\b	バックスペース (BS)
\f	フォームフィード (FF)
\n	行送り (LF)
\r	復帰 (CR)
\t	水平タブ (TAB)
\uhhhh	16-bit の 16 進数値 <i>hhhh</i> を持つ Unicode 文字
\Uhhhhhhhh	32-bit の 16 進数値 <i>hhhhhhhh</i> を持つ Unicode 文字 (Field Reports での拡張仕様)

5.1.2 Python から利用する場合

Python から Field Reports を利用する場合は、基本データ型と Python のデータ型を表 5.4 のように対応付けて、レンダリングパラメータとなるデータ構造を作成します。

表 5.4: Python データ型との対応

基本データ型	Python データ型
数値	整数 (int 型) または浮動小数点数 (float 型)
真理値	真偽値 (bool 型)
文字列	文字列 (string 型) または Unicode 文字列 (unicode string 型)
列挙値	文字列 (string 型) または Unicode 文字列 (unicode string 型)
辞書	辞書 (dict 型)
リスト	リスト (list 型) またはタプル (tuple 型)

辞書のキーとして、文字列 (string 型) または Unicode 文字列 (unicode string 型) が使用できます。

文字列 (string 型) の文字コードは、UTF-8 としてください。

5.1.3 Ruby から利用する場合

Ruby から Field Reports を利用する場合は、基本データ型と Ruby のデータ型を表 5.5 のように対応付けて、レンダリングパラメータとなるデータ構造を作成します。

表 5.5: Ruby データ型との対応

基本データ型	Ruby データ型
数値	整数 (Fixnum または Bignum) または浮動小数点数 (Float)
真理値	true または false
文字列	文字列 (String)
列挙値	文字列 (String) またはシンボル (Symbol)
辞書	ハッシュ (Hash)
リスト	配列 (Array)

ハッシュのキーとして、文字列 (String) またはシンボル (Symbol) が使用できます。

Ruby1.8 以前の場合、文字列の文字コードを UTF-8 としてください。

Ruby1.9 以降の場合は、文字列自身が持つ Encoding と文字コードが一致しているものとします。

5.1.4 PHP から利用する場合

PHP から Field Reports を利用する場合は、基本データ型と PHP のデータ型を表 5.6 のように対応付けて、レンダリングパラメータとなるデータ構造を作成します。

表 5.6: PHP データ型との対応

基本データ型	PHP データ型
数値	数値
真理値	論理型
文字列	文字列
列挙値	文字列
辞書	連想配列 (配列要素がキーを持つ)
リスト	添字配列 (配列要素がキーを持たない)

文字列の文字コードは、UTF-8 としてください。

配列中にキーを持たない要素とキーを持つ要素が混在している場合は、数字のキーを持つ連想配列とみなします。

5.2 共通データ構造

基本データ構造を組み合わせて構築される汎用のデータ構造がいくつかあり、レンダリングパラメータの各所で使用されます。

この節では、長さ・比率・日付／時刻・URL・色のデータ構造について説明します。

5.2.1 長さ

長さを指定する場合、数値または単位付き数値の文字列表現が利用できます。数値で長さを指定する場合は、特に注記のないかぎりポイント (pt) 単位となります。

表 5.7: 長さの単位一覧

単位	説明
mm	ミリメートル
cm	センチメートル
in	インチ (1in = 2.54cm)
pt	ポイント (1pt = 1/72in=0.3528mm)
pc	パイカ (1pc = 12pt)
q	級 (1q = 0.709pt)

5.2.2 比率

比率を指定する場合、単位記号 “%” 付き数値の文字列表現として百分率（パーセント）で指定するか、0 から 1 の数値で指定します (1.0 = "100%")。

5.2.3 日付／時刻

日付または時刻は、以下の書式の文字列で表現します (ISO 8601 のサブセット)。

日付 → YYYY[-MM[-DD]]

時刻 → hh[:mm[:ss]]]

日付と時刻 → YYYY[-MM[-DD]]T hh[:mm[:ss]]]

日付と時刻を同時に表記する場合は、日付と時刻を区切り記号 “T” または空白文字 “_” で区切ります。

5.2.4 色

数値配列による指定

色指定が必要な場面では、数値のリストにより色成分を指定します。

透明色 → []

グレースケール色 → [数値] または 数値

RGB 色 → [数値 , 数値 , 数値]

CMYK 色 → [数値 , 数値 , 数値 , 数値]

グレースケール色では、0~255 の 1 要素数値リストまたは数値で階調を表現します。RGB 色では、赤・緑・青の各色成分を 0~255 の 3 要素の数値リストとして表現します。CMYK 色では、シアン・マゼンタ・黄・黒の各色成分を 0~255 の 4 要素の数値リストとして表現します。

色名による指定

色名による指定では、表 5.8 にあげる列挙値が使用できます。

表 5.8: 色名の一覧

列挙値	値
Transparent (透明)	[]
Black	[0, 0, 0]
Gray	[128, 128, 128]
Silver	[192, 192, 129]
White	[255, 255, 255]
Maroon	[128, 0, 0]
Red	[255, 0, 0]
Purple	[128, 0, 128]
Fuchsia	[255, 0, 255]
Green	[0, 128, 0]
Lime	[0, 255, 0]
Olive	[128, 128, 0]
Yellow	[255, 255, 0]
Navy	[0, 0, 128]
Blue	[0, 0, 255]
Teal	[0, 128, 128]
Aqua	[0, 255, 255]

CSS 形式文字列による指定

以下の書式の文字列による色指定も可能です。

RGB 色 → # 6 桁 16 進数値

色名 → 列挙値

RGB 色 → **rgb(r, g, b)**

CMYK 色 → **cmyk(c, m, y, k)**

rgb(r,g,b) 形式の場合、色成分値を数値 (0~255) または百分率で指定します。

cmky(c,m,y,k) 形式の場合、色成分値を数値 (0~1) または百分率で指定します。

5.2.5 URL

PDF ファイル・画像ファイル・フォントファイルなどのファイルの場所を指定する場面では、URL によりリソースの場所を指定します。

URL は、以下の書式の文字列です。

URL → ジークーム名 + ドメイン名 + パス名

data URI scheme 文字列

以下の data URI scheme 文字列形式を用いて、ファイルの内容をレンダリングパラメータにインラインで埋めこむことができます。

data URI scheme 文字列 → **data:** *MIME-type* [;**base64**], データ列

MIME-type として指定できる値は以下のとおりです。

- application/pdf
- image/jpeg
- image/jp2
- image/png
- image/x-bmp

“;base64” が付いている場合は、カンマ以降のデータ列 Base64 デコードしてから取り込みます。“;base64” を省略した場合は、カンマ以降にバイナリデータが続いているものとします。

文字列の終端をヌル文字で判断する C 言語タイプの文字列データを使用しているプログラミング言語では、文字列にバイナリデータを埋め込むことができませんので、Base64 エンコードが必須となります。また、JSON で記述する場合も Base64 エンコードが必要です。

ローカルファイル

スキーム名・ドメイン名を省略しパス名のみを記述した場合は、ローカルファイルを指示していると解釈します。

ローカルファイルのパス名が “/” もしくはドライブ名（Windows のみ）で始まる場合は、絶対パスによるファイルの指定として扱います。

パス名が “.” もしくは “..” で始まる場合は、カレントディレクトリ相対のパス名によるファイル指定とみなします。Field Reports プロセスのカレントディレクトリを基準として、ファイルを指定します。

絶対パス形式でなく、かつ先頭が ‘.’ や ‘..’ で始まらない場合は、template-root (5.12) 相対のパス名によるファイル指定として扱います。

5.3 セレクタ文字列

ピリオド “.” を区切り文字として部分セレクタを結合したものをセレクタと呼びます。

部分セレクタは、名前セレクタ・全称セレクタ・整数セレクタのいずれかです。名前セレクタは、リテラル文字列で指定し、同じ文字列を持つ部分フィールド名とマッチします。全称セレクタは、 “*” で指定し、任意の部分フィールド名とマッチします。整数セレクタは、開始値、終了値、ステップ数 から生成される整数列のいずれかと同じ値を持つ部分フィールド名とマッチします。

セレクタと完全修飾フィールド名とのマッチング処理では、ルートの部分フィールド名から順にマッチングを試みていきます。途中でマッチングが失敗するか、すべての部分セレクタのマッチングが成功した時点で、マッチング処理は終了します。マッチング処理全体が成功した場合は、最後に検査したフィールドとその子フィールドすべてが選択対象となります。

以下にセレクタ文字列の書式を示します。

```
セレクタ → 部分セレクタ . . .
部分セレクタ → 名前セレクタ
| 全称セレクタ
| 整数セレクタ
名前セレクタ → リテラル文字列
全称セレクタ → *
整数セレクタ → [ インデックス値 ]
→ [[開始値]:[終了値]]
→ [[開始値]:[終了値]:ステップ数 ]
```

整数セレクタの第一の書式では、インデックス値により要素を一つ選択します。第二・第三の書式では、開始値から終了値までの範囲の整数列にマッチする要素を選択します。ただし第三の書式では、整数列を生成する際にステップ数づつ加算していきます。

開始値・終了値はそれぞれ省略可能です。開始値を省略した場合は 0 と解釈します。終了値を省略した場合は、最後の要素までを範囲とします。

整数セレクタでは、図 5.1 のように、要素と要素の間にインデックスがあると考えます。終了値にマイナスの数値を使用した場合は、最後の要素から数えて何個目かを示します。

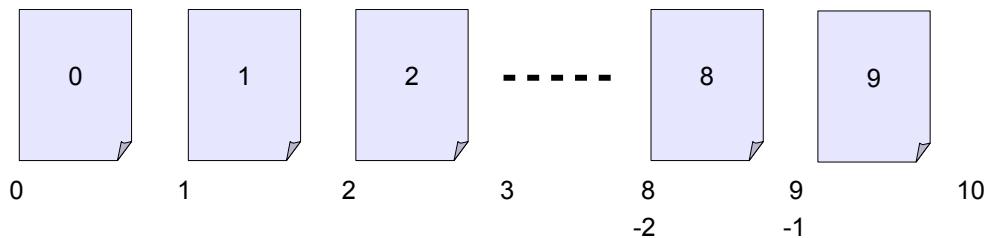


図 5.1: 開始値と終了値の考え方

整数セレクタの記述例を表 5.9 にいくつか示します（要素が 10 個の場合）。

表 5.9: 整数セレクタの使用例

名前パターン	意味
[1]	1 にマッチする。
[-2]	8 にマッチする。
[0:2]	0,1 にマッチする。
[0:-1]	0,1,2,...,8 にマッチする。
[0:10]	0,1,2,...,9 にマッチする。
[::3]	0,1,2 にマッチする。
[8:]	8,9 にマッチする。
[:]	0,1,2,...,9 にマッチする。
[1::2]	1,3,5,7,9 にマッチする。

5.4 レンダリング辞書

レンダリングパラメータとして、表 5.10 に示すレンダリング辞書を与えます。

表 5.10: レンダリング辞書のエントリ

キー	型	値
template	template 要素	(必須) 名前空間を定義し、PDF テンプレートと関連付けます。 複数の PDF テンプレートを連結して PDF 帳票を組み立てる際には、PDF テンプレートに元々存在するフィールドの名前とここで定義した名前空間を合わせて、新しいフィールド名とします。
resources	resources 辞書	フォント・画像リソースを定義します。 リソースとして定義すると、データ読み込み時・PDF 出力時に処理の重複を避けることができるるので、処理時間とファイルサイズの節約になります。
context	context 要素	フィールド名に対応する属性を 1 対 1 対応で指定します。 フィールドが階層構造を持つ場合は、辞書とリストを使って、相似形の木構造データとして記述します。
style	style リスト	複数のフィールドに対応する属性をスタイルにより一括して指定します。 セレクタを用いたパターンマッチングにより、適用対象となるフィールドを特定します。
environ	environ 辞書	ユーザー定義の環境変数を定義します。 フィールド値の一部に “\${ 環境変数名 }” の並びが出現する箇所で、環境変数の値に置き換えられます。ページ番号・全ページ数などのシステム定義の環境変数については 5.11.2 を参照してください。
property	property 辞書	生成される帳票の PDF 属性を指定します。 文書情報・セキュリティ・文書の開きかたなどの属性を設定することができます。
settings	settings 辞書	Field Reports の共通設定情報を格納します。 シリアル番号、ライセンスキー、PDF テンプレート格納ディレクトリなどの情報を記述します。

フィールド属性に影響を与えるレンダリングパラメータは、以下の順序で適用されます。

1. style リスト
2. context 要素
3. 環境変数

style リストと context 要素の指定が重複した場合は、context 要素による指定が優先されます。
環境変数の展開は、style リストと context 要素の適用後の value 属性の値に対して行われます。

5.5 template 要素

帳票で使用する PDF テンプレートを宣言します。

ここでは名前空間を定義し、PDF 要素との対応付けを行ないます。template 要素の書式を表 5.11 に示します。

表 5.11: template 要素の書式

書式	説明
PDF 要素	名前空間が不要な場合の書式です。 単票で使用します。
[{ 名前空間 : PDF 要素 } , ...]	複合帳票または連続帳票を構成する場合の書式です。 Standard エディション以上で利用できます。

5.5.1 名前空間

複合帳票の名前空間

部分フィールド名を“.”で連結して、名前空間を構成します。

名前空間 → 部分フィールド名
| 部分フィールド名 . 名前空間

連続帳票の名前空間

同じく部分フィールド名を“.”で連結しますが、最後は特殊フィールド名“*”とします。

名前空間 → *
| 部分フィールド名 . 名前空間

5.5.2 PDF 要素

PDF 要素の書式は、表 5.12 のとおりです。

表 5.12: PDF 要素の書式

書式	説明
<i>URL</i>	<i>src</i> 属性の指定だけでよい場合の省略記法です。
PDF 辞書	<i>src</i> 以外の属性も指定する場合は辞書形式とします（表 5.12 参照）。
[PDF 辞書 , ...]	連続帳票で条件付きテンプレート選択を行う場合は、辞書のリストとします。

単に PDF ファイルの URL を指定するだけで十分な場合は、 URL を直接記述します。その他の属性を指定する必要がある場合は、 PDF 辞書を記述します。連続帳票において条件付きテンプレート選択を行う場合には、 PDF 辞書を要素とするリストを記述します。

PDF 辞書は、表 5.13 のエントリを持つ辞書です。

表 5.13: PDF 辞書のエントリ

キー	型	値
src	URL	(paper を指定しない場合は必須) PDF テンプレートの URL を指定します。
paper	長さリスト, 列挙値	(src を指定しない場合は必須) PDF テンプレートを使用しないで、白紙のページを新規に生成します。白紙のページに動的に生成したテキスト・画像フィールドを配置する場合に使用します。ここで指定した用紙サイズは、出力する PDF の MediaBox (用紙サイズ) に反映されます。数値リスト形式で指定する場合は、幅・高さを 2 要素のリストとしてポイント単位で指定します。列挙値で指定する場合は、次の選択値のどれかを指定します。選択値: A0, A1, ... A10, B0, B1, ... B10, Letter, Legal.
orientation	列挙値	用紙の方向を指定します。paper と合わせて使用します。選択値: Portrait (縦長), Landscape (横長). 初期値: Portrait.
crop-box	長さリスト	出力する PDF の CropBox (トリミング範囲) を指定します。MediaBox の左下を原点とした座標系における左下・右上の座標を 4 要素の数値リストとして与えます。書式: [<i>llx</i> , <i>lly</i> , <i>urx</i> , <i>ury</i>].
bleed-box	長さリスト	出力する PDF の BleedBox (裁ち落としサイズ) を指定します。書式は同上。
trim-box	長さリスト	出力する PDF の TrimBox (仕上がりサイズ) を指定します。書式は同上。
art-box	長さリスト	出力する PDF の ArtBox (アートサイズ) を指定します。書式は同上。
rows	整数, 辞書	(可変テーブルでない連続帳票では必須) テーブル形式フィールドの最大の行数を指定します。テンプレート内にテーブルがひとつだけ、もしくはすべてのテーブルの行数が同一の場合は、単に整数で指定します。複数のテーブルが存在し、かつそれぞれの行数が異なる場合は、それぞれのテーブルの行数を辞書形式で個別に指定します (書式: {部分フィールド名 : 最大行数, ...})。rows エントリの値を元にテーブル分割処理を行います。テーブル行数が 0 の場合、テーブル分割処理は行いません。初期値: 0.
pages	整数リスト	PDF テンプレートとして実際に使用するページを列挙します。ページ数は 0 始まりです。初期値: [] (全ページ).
match	文字列	連続帳票において自動生成される名前空間 “0,1,2,...” と特定のテンプレートを対応付ける条件を部分セレクタ文字列として記述します (3.2.5)。初期値: “*”
exclude-annotations	列挙値リスト	PDF テンプレートから指定した種類の注釈を除きます。 選択値: Text, Link, FreeText, Line, Square, Circle, Highlight, Underline, StrikeOut, Stamp, Ink, Popup, FileAttachment, Sound, Movie, Widget, TrapNet, Polygon, PolyLine, Squiggly, Caret, Screen, PrinterMark, Watermark, 3D, All (すべての注釈を除く) . 初期値: [] (すべての注釈を残す).

条件付きテンプレート選択

match エントリでは、 “*” または “.*” で終わる名称として定義した名前空間の “*” の部分を選択する条件を部分セレクタ文字列として記述します。ただし、整数セレクタの終了値としてマイナスの数値は利用できません。

実際にテンプレートを選択する際には、現在処理中のページの名前空間をパラメータとして、PDF 辞書の match エントリの選択条件をリスト要素の格納順に検査していきます。選択条件が真となる PDF 辞書が見つかった時点で検査を打ち切り、その PDF 辞書を選択します。選択条件に当てはまる PDF 辞書が見つからなかった場合は、エラー終了します。

5.6 resources 辞書

繰り返し使用するフォント・画像データをリソースとして定義します。

表 5.14: resources 辞書のエントリ

キー	型	値
font	font 要素	フォント・リソースを定義します。
image	image 要素	画像リソースを定義します。

5.6.1 font 要素

表 5.15 に示す font リソース辞書で、フォントリソースを定義します。

表 5.15: font 辞書のエントリ

キー名	型	値
フォント・リソース名	URL, font 属性辞書	フォントファイルの URL または font 属性辞書

URL 以外の詳細なフォント属性を指定する場合は、表 5.16 に示す font 属性辞書を記述します。

表 5.16: font 属性辞書のエントリ

キー名	型	値
src	URL	(必須) フォントファイルの URL を指定します。
embed	真偽値	フォントを PDF に埋め込むかどうかの指定をします。初期値 : true.
subset	真偽値	フォントを PDF に埋め込む際にサブセット化を行うかどうかの指定をします。 embed が true の場合のみ有効です。初期値 : true.
ttc-index	整数	TTC フォントのインデックス番号を指定します。TTC フォントの場合のみ有効です。初期値 : 0.
writing-mode	列挙値, 整数	テキストの組み方向を規定します。 HorizontalTb または 0 横組み。行送り方向は上から下。 VerticalRl または 1 縦組み。行送り方向は右から左。 初期値 : HorizontalTb.
text-orientation (2.0)	列挙値	writing-mode が縦組みの場合に、英数文字の向きを指定します。 Upright 英数文字を自然な向きで、1 文字1 文字配置します。 Sideways 英数文字を時計回りに 90 度回転させて配置します。 初期値 : Upright.
script (2.0)	文字列リスト	GSUB テーブルを選択する際に利用するスクリプトを指定します。フォントに GSUB テーブルが存在しない場合は機能しません。初期値 : ["DFLT", "kana"].
language (2.0)	文字列リスト	GSUB テーブルを選択する際に利用する言語を指定します。フォントに GSUB テーブルが存在しない場合は機能しません。初期値 : ["dflt"].
features (2.0)	文字列リスト	GSUB テーブルを選択する際に利用するフィーチャーを指定します。フォントに GSUB テーブルが存在しない場合は機能しません。初期値 : [].
baseline-shift (2.0)	長さ	フォントのベースライン位置を調整します。初期値 : 0.

縦組み時の features エントリ

縦組みの場合、長音記号「ー」等の一部記号（役物）を縦組み用のものに切り替える必要があります。Field Reports では、フォントに組み込まれた GSUB テーブルの選択によりこの切替えを実現しています。

writing-mode で縦組みが指定され text-orientation が Upright の場合、features エントリに “vert” を追加します。 Sideways の場合は、features エントリに “vrt2” を追加します。

横組みと縦組みを同時に使用する場合のフォント定義

一つのフォントファイルを縦組みと横組みで同時に使用する場合は、フォントリソース名と writing-mode を変えたリソースを 2つ定義してください。

```
{  
  "resources": {  
    "font": {  
      "HiraMaruPro-W4": {  
        "src": "/usr/lib/fonts/HiraMaruGo_Pro_W4.otf",  
        "writing-mode": "HorizontalTb"  
      },  
      "@HiraMaruPro-W4": {  
        "src": "/usr/lib/fonts/HiraMaruGo_Pro_W4.otf",  
        "writing-mode": "VerticalRl"  
      }  
    }  
  }  
}
```

コード 5.1: 横組みと縦組みを同時に使う場合の設定例

TTC フォントを使用する場合のヒント

TTC フォントとは、複数の TrueType フォントを一つのファイルに格納した形式のフォントです。Field Reports では、0 始まりの ttc-index により TTC フォント内のフォントを識別します。

reports コマンドの “font” サブコマンドを使うと、フォントファイルの内部情報が調べられます。このコマンドにより、TTC フォントが内蔵しているフォントの数を知ることができます。

【実行例】

```
$ reports font msmin04.ttc  
[msmin04.ttc]  
type: TrueType  
PostScript name: MS-Mincho  
ttc index: 0  
  
[msmin04.ttc]  
type: TrueType  
PostScript name: MS-PMIncho  
ttc index: 1
```

フォントリソースの雛形の作成方法

“font” サブコマンドに “—json” オプションを付けて実行することで、フォントファイルからフォントリソース定義の雛形を生成することができます。

【実行例】

```
$ reports font --json msmin04.ttc
{
  "resources": {
    "font": {
      "MS-Mincho": {
        "src": "msmin04.ttc",
        "embed": true,
        "subset": true,
        "ttc-index": 0,
        "writing-mode": 0
      },
      "MS-PMincho": {
        "src": "msmin04.ttc",
        "embed": true,
        "subset": true,
        "ttc-index": 1,
        "writing-mode": 0
      }
    }
  }
}
```

既存フォントの置き換え

PDF テンプレートで使用しているフォントと同じ名前のフォントリソースを定義した場合、font 辞書で定義したフォントが優先して使用されます。この動作を利用して、PDF テンプレートで設定した既存フォントの内容を置き換えることができます。このテクニックは、フォントを埋め込む指定を一括して行う場合に便利です。

PDF テンプレートの各フィールドに対して設定されているフォントの正確な内部名称は、reports コマンド（[8.6 参照](#)）の “parse” サブコマンドにより確認することができます（以下の実行例の場合、“title” フィールドに設定されているフォントは “/KozMinProVI-Regular” であることがわかる）。

【実行例】

```
$ reports parse template.pdf
...
"title": {
    "ftype": "Tx",
    "rect": [ 150.788, 625.573, 300.311, 643.103 ],
    "font": "/KozMinProVI-Regular",
    "font-size": 10.0,
    "color": [ 0 ],
    "text-stroke-color": [],
    "value": "",
    "multiline": false,
    "text-align": "Left",
    "border-color": [],
    "background-color": [],
    "rotate": 0
},
...
...
```

5.6.2 image リソース辞書

image リソース辞書の書式は、表 5.17 のとおりです。

表 5.17: image リソース辞書のエントリ

キー名	型	値
画像リソース名	URL, image 属性辞書	画像ファイルの URL または image 属性辞書

image 辞書要素のキーとして任意の画像リソース名を定義します。ここで定義したリソース名は、ボタン(画像)フィールド属性 (5.9.9) で表示する画像を指定する際に利用します。

image 辞書要素の値として、URL または表 5.18 に示す image 属性辞書を与えます。単に画像ファイルの URL を指定するだけ十分な場合は、画像ファイルの URL を文字列で指定します。その他の画像属性も指定する必要がある場合は、image 属性辞書を使用します。

表 5.18: image 属性辞書のエントリ

キー名	型	値
src	URL	(必須) 画像ファイルの URL を指定します。
page (廃止予定)	整数値	画像ファイルとして PDF を指定する際に使用するページを指定します。PDF 形式の場合のみ有効なエントリです。最初のページ番号は 0 です。初期値 : 0 ※複数ページの画像から特定ページを参照する際には、ページ番号付きのリソース名 (5.9.9) を利用してください。
enable-bmp-alpha	真偽値	32 ビットモード BMP 形式の画像を指定した際に、RGB (24 ビット) 以外の残りの 8 ビットを α チャンネルとして使用するかどうかを指定します。32 ビット BMP 形式の場合のみ有効なエントリです。初期値 : false.
transparent-range	整数リスト	透明色として扱う色の範囲を指定します。アルファチャンネルを持たない PNG, BMP 画像形式で有効なエントリです。 $2 \times n$ 個の整数リスト $[min_1, max_1, \dots, min_n, max_n]$ で指定します。ラスタ画像の色モードが RGB カラーであれば $n = 3$, グレースケールであれば $n = 1$ となります。色の範囲はラスタ画像のビット深度 d に依存し, $0 \sim 2^d - 1$ となります。例えば RGB24 ビットカラー画像で赤を透過色にするには, $[255, 255, 0, 0, 0, 0]$ とします。インデックスカラーの場合は, $n = 1$ として 2 つのインデックス番号値で色の範囲を指定します。初期値 : [].

対応画像形式

image 属性で利用可能な画像ファイルの形式は、表 [5.19](#) のとおりです。

表 5.19: 対応している画像形式

画像形式	拡張子	Media Type	制限事項
JPEG	*.jpg, *.jpeg	image/jpeg	・特になし。
JPEG2000	.*.jp2	image/jpeg	・特になし。
PNG	*.png	image/png	・ビット深度が 8 ビット未満となる色モードには対応していません。 ・インターレースモードには対応していません。
BMP	*.bmp	image/x-bmp	・透過色の扱いについては、enable-bmp-alpha の項を参照してください。
PDF	*.pdf	application/pdf	・セキュリティが有効な PDF は使用できません。 ・ページ指定の方法については表 5.18 を参照してください。

5.7 context 要素

context 要素では、部分フィールド名とフィールド属性の組を列挙します。辞書とリストを組み合わせて、フィールドの階層構造に一致する木構造のデータを組み立てます。

context 要素の書式を表 [5.20](#) に示します。

表 5.20: context 要素の書式

書式	説明
{ 部分フィールド名 : (<i>field</i> 要素 <i>context</i> 要素) }	辞書形式で記述する場合： 部分フィールド名 が末端のフィールド名の場合, <i>field</i> 要素 を与えます。部分フィールド名 が末端のフィールド名でない場合, <i>context</i> 要素 を再帰的に与えます。
[(<i>context</i> 要素 <i>field</i> 要素), ...]	リスト形式で記述する場合： リストが, 1次元テーブルにおける「行」, 2次元テーブルにおける「列」の位置にある場合, <i>field</i> 要素 を与えます。それ以外では, <i>context</i> 要素 を再帰的に与えます。

辞書形式では, 部分フィールド名をキーに, *field* 要素 (5.9) または *context* 要素を値として与えます。

部分フィールド名が 0 始まりの整数列の場合に限り, リスト形式で記述することができます。これは, 「{"0": *context* 要素₀, "1": *context* 要素₁, ... }」の形式の辞書を略した記法とみなすことができます。リスト形式の表記はまた, テーブル分割機能において, 分割ポイントを探すための目印にもなります。テーブル分割機能を利用する場合は, 「行」位置のデータをリスト形式で記述してください。

5.8 style リスト

style を適用するフィールドをセレクタで指定します。

表 5.21: style リストの書式

書式	説明
[{ セレクタ : <i>field</i> 要素 }, ...]	リストの要素として, セレクタ文字列と <i>field</i> 要素の組を列挙します。リストの先頭要素から順にセレクタにマッチするフィールドを探していく, マッチしたフィールドに対して <i>field</i> 要素を適用します。

セレクタの表記法については, 5.3 を参照してください。

field 要素の書式は, *context* 要素で説明した *field* 要素と同じです。5.9 を参照してください。ただし, style リストで new 属性を指定して, 新規のフィールド生成を指示することはできません。

5.9 field 要素

field 要素の書式は, 表 5.22 のとおりです。

表 5.22: field 要素の書式

フィールド属性	説明
リテラル値 (文字列, 数値, 真理値)	<i>field</i> 要素として value 属性のみで十分な場合の略記法です。
<i>field</i> 辞書	value 以外の属性を指定する場合の書式です。

field 辞書に指定可能な属性は、フィールドの種類ごとに異なります（表 5.23）。

表 5.23: フィールド属性一覧

フィールド属性区分	テキスト	ボタン	可変テーブル	セル
共通フィールド属性（5.9.2）	○	○	○	-
コンテンツ・スタイル属性（5.9.3）	○	○	○	○
境界線スタイル属性（5.9.4）	○	○	○	○
テキスト・バリュー属性（5.9.5）	○	-	○	○
テキスト・スタイル属性（5.9.6）	○	-	○	○
テキスト・レイアウト属性（5.9.7）	○	-	○	○
テキスト改行属性（5.9.8）	○	-	○	○
ボタンフィールド属性（5.9.9）	-	○	-	-
可変テーブル属性（5.9.10）	-	-	○	-

5.9.1 フィールド属性

以降の項でフィールド属性を例挙します。

表中、拡張欄が○の項目は拡張属性を示します。拡張属性は permanent 属性が true の場合のみ使用してください。

表中の「初期値」は、new 属性を指定して新規にフィールドを作成した場合の初期値もしくは拡張属性の初期値を示します。非拡張属性の初期値は、PDF テンプレートで設定された属性値となります。

5.9.2 共通フィールド属性

表 5.24 は、フィールド全体のスタイルを指定するための属性です。

表 5.24: 共通フィールド属性

キー	拡張	型	値
new または ftype (2.0)	-	列挙値	(新規作成時必須) 新規に作成するフィールドの種類を指定します。 Tx テキストフィールド Btn ボタン (画像) フィールド Tbl (2.0) 可変テーブル ここで指定した種類のフィールドを動的に生成し、帳票に出力します。
rect	-	長さリスト	(新規作成時必須) フィールドの矩形座標を指定します。ページの左下を原点として、左下・右上座標を 4 要素のリストとして与えます。書式：[llx , lly , urx , ury].
llx	-	長さ	フィールドの X 座標を変更します。
lly	-	長さ	フィールドの Y 座標を変更します。
width	-	長さ	フィールドの幅を変更します。
height	-	長さ	フィールドの高さを変更します。
translation	-	リスト	フィールドの位置を相対的に移動させます。現在のフィールドの座標からの移動量を 2 要素のリストとして与えます。単位：ポイント。書式：[dx , dy].
rotation	○	数値, 数値リスト	フィールドの回転角度を指定します。angle と異なり境界線を含めたフィールド全体が回転します。単一の数値で指定した場合、回転の中心はフィールドの左下となります。3 要素の数値リストで指定した場合は、回転の中心位置をフィールドの左下からの相対座標で指定します。単位：角度 (度数法)。書式：角度 または [X 座標 , Y 座標 , 角度].
matrix	○	6 要素数値リスト	座標変換行列を指定します。境界線を含めたフィールド全体に対してアフィン変換を掛けます。書式：[a , b , c , d , e , f].
pages	○	整数, 整数リスト	new 属性を指定してフィールドを配置する際に、対象となるページを指定します。テンプレートが複数ページの場合に意味のある属性です。テンプレート内でのページ数を 0 始まりで指定します。初期値：[] (全ページ).
clipping	○	真理値	フィールドからはみ出した内容物をクリッピングします。初期値：true.
clip-path (2.0)	○	文字列	フィールドのクリッピング範囲をパスで記述します。書式は、path 要素の d 属性 (7.1.8) と同様ですが、フィールドの左下を原点とし y 軸の値が上方向に大きくなる座標系となります。
opacity	○	比率	フィールドの不透明度を 0~1 の数値で指定します。0 で完全に透明になります。書式：不透明度または [ストローク色不透明度, 非ストローク色不透明度]. 初期値：1.
blend-mode	○	列挙値	フィールドを背景と合成する際のブレンドモードを指定します。選択値：Normal, Multiply, Screen, Overlay, Darken, Lighten, ColorDodge, ColorBurn, HardLight, SoftLight, Difference, Exclusion, Hue, Saturation, Color, Luminosity.
permanent	○	真理値	true の場合、該当フィールドをコンテンツストリームに変換してテレプレートの外観に重畠します。false の場合は、フィールドのまま残します。初期値：true.

(続く)

(続き)

キー	拡張	型	値
visible	-	真理値	フィールドの表示／非表示を切り替えます。初期値：true.
action (2.0)	-	アクションリスト	フィールドをクリックした時に実行するアクションを指定します。permanent 属性が false の場合のみ有効です。初期値：[].

PDF 座標系

フィールドの座標は、PDF 座標系で与えます（図 5.2）。PDF 座標系は、用紙の左下を原点とし、x が右方向、y が上方向に大きくなります。

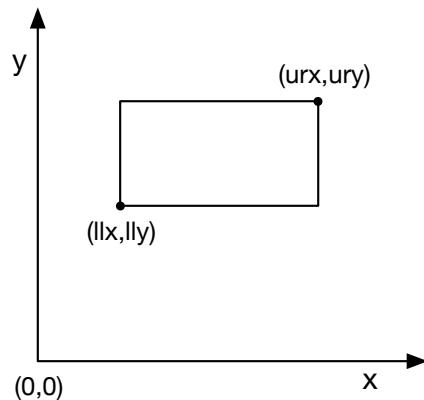


図 5.2: PDF 座標系

アクション

アクション辞書により、アクションの種類と動作を指定します（表 5.25）。

アクションの種類として、現状 JavaScript のみ対応しています。

表 5.25: アクション辞書のエントリ

キー名	型	値
type	列挙値	（必須）アクションの種類指定します。固定値：JavaScript.
js	JavaScript 文字列	（必須）JavaScript 文字列を指定します。

```
{
  "template": {"paper": "A4"},
  "context": {
    "hello": {
      "new": "Btn",
      "border-width": 1,
      "background-color": "Blue",
      "rect": [100, 700, 230, 750],
      "permanent": false,
      "action": [
        {"type": "JavaScript", "js": "app.alert(\"Adobe Acrobat #1\");"},
        {"type": "JavaScript", "js": "app.alert(\"Adobe Acrobat #2\");"}
      ]
    }
  }
}
```

コード 5.2: JavaScript アクションの記述例

5.9.3 コンテンツ・スタイル属性

表 5.26 は、フィールドの中身のスタイルを指定するための属性です。

表 5.26: コンテンツ・スタイル属性

キー	拡張	型	値
padding	○	長さ, 長さリスト	境界線と内容物の間の余白の量を指定します。長さを指定した場合、左・下・右・上に同じ余白を指定します。2要素のリスト形式で指定した場合、左右・上下の順に余白を指定します。4要素のリスト形式で指定した場合、左・下・右・上の順に余白を指定します。書式: [左, 下, 右, 上]. 初期値: border-style が Beveled または Inset の時は 2, それ以外は 1.
background-color	-	色	フィールドの背景色を指定します。初期値: [] (背景色なし).
angle	-	角度	フィールドの中身の描画方向を指定します。Adobe Acrobat でフィールドのプロパティを編集する際の「向き」に対応します。90 度単位の角度を設定してください (0, 90, 180, 270).

5.9.4 境界線スタイル属性

表 5.27 は、フィールド境界線のスタイルを指定するための属性です。

表 5.27: 境界線スタイル属性

キー	拡張	型	値
border-width	-	長さ	境界線の太さを指定します。0を指定すると、境界線が表示されません。単位：ポイント。新規作成時の初期値：0.
border-left-width (2.0)	○	長さ	左境界線の太さを指定します。
border-bottom-width (2.0)	○	長さ	下境界線の太さを指定します。
border-right-width (2.0)	○	長さ	右境界線の太さを指定します。
border-top-width (2.0)	○	長さ	上境界線の太さを指定します。
border-color	-	色指定	境界線の表示色を指定します。初期値：[0]（黒）。
border-left-color (2.0)	○	色指定	左境界線の表示色を指定します。
border-bottom-color (2.0)	○	色指定	下境界線の表示色を指定します。
border-right-color (2.0)	○	色指定	右境界線の表示色を指定します。
border-top-color (2.0)	○	色指定	上境界線の表示色を指定します。
border-style	-	列挙値	境界線のスタイルを指定します。選択値：Solid, Dashed, Beveled, Inset, Underline. 初期値：Solid.
border-left-style (2.0)	○	列挙値	左境界線のスタイルを指定します。
border-bottom-style (2.0)	○	列挙値	下境界線のスタイルを指定します。
border-right-style (2.0)	○	列挙値	右境界線のスタイルを指定します。
border-top-style (2.0)	○	列挙値	上境界線のスタイルを指定します。
border-dash	-	長さリスト	border-style が Dashed の場合に、破線パターンを指定します。交互に現れる破線と隙間の長さを2要素までの数値リストとして指定します。1要素の場合、破線と隙間の長さは同じになります。2要素の場合、破線・隙間の順にそれぞれの長さを指定します。初期値：[3].
border-left-dash (2.0)	○	長さリスト	左境界線の破線パターンを指定します。
border-bottom-dash (2.0)	○	長さリスト	下境界線の破線パターンを指定します。
border-right-dash (2.0)	○	長さリスト	右境界線の破線パターンを指定します。
border-top-dash (2.0)	○	長さリスト	上境界線の破線パターンを指定します。
border	-	境界線スタイル辞書	境界線のスタイルをまとめて指定します。境界線スタイル辞書（表 5.28）を指定できます。
border-left (2.0)	○	境界線スタイル辞書	左境界線のスタイルをまとめて指定します。
border-bottom (2.0)	○	境界線スタイル辞書	下境界線のスタイルをまとめて指定します。
border-right (2.0)	○	境界線スタイル辞書	右境界線のスタイルをまとめて指定します。
border-top (2.0)	○	境界線スタイル辞書	上境界線のスタイルをまとめて指定します。

(続く)

(続き)

キー	拡張	型	値
border-join-style	○	列挙値	境界線の角のスタイルを指定します。 MiterJoin 境界線の角を直角に描きます。 RoundJoin 境界線の角を丸めます。 BevelJoin 境界線の角を切り落とします。 初期値 : MiterJoin.
border-radius (2.0)	○	長さリスト	境界線の角の丸めを指定します。1要素の場合、角の丸めに用いる円の半径を指定します。2要素の場合、角の丸めに用いる楕円のX軸半径とY軸半径を指定します。border-style が Solid または Dashed であり、かつ上下左右の境界線スタイルが等しい場合にのみ有効です。初期値 : [0, 0].

境界線のスタイル

border-style により、境界線のスタイルは図 5.3 のように変化します。



図 5.3: 境界線のスタイル

境界線の角のスタイル

border-join-style により、境界線の接続部分のスタイルは図 5.4 のように変化します。



図 5.4: 境界線の角のスタイル

境界線スタイル辞書

境界線の太さ・色・スタイル・破線パターンをまとめて指定する場合に使用します。

表 5.28: 境界線スタイル辞書

キー	型	値
width	長さ	境界線の太さを指定します。0 を指定すると、境界線が表示されません。単位：ポイント。新規作成時の初期値：0.
color	色指定	境界線の表示色を指定します。初期値：[0]（黒）。
style	列挙値	境界線のスタイルを指定します。選択値：Solid, Dashed, Beveled, Inset, Underline。初期値：Solid。
dash	長さリスト	border-style が Dashed の場合に、破線パターンを指定します。交互に現れる破線と隙間の長さを 2 要素までの数値リストとして指定します。1 要素の場合、破線と隙間の長さは同じになります。2 要素の場合、破線・隙間の順にそれぞれの長さを指定します。初期値：[3]。

5.9.5 テキスト・バリュー属性

テキストの値を取るフィールドにおいて、以下の表に示す属性を設定することができます。

表 5.29: テキスト・バリュー属性

キー	拡張	型	値
value	-	文字列・数値 または真理値	テキストフィールドの値を指定します。 値が数値または真理値の場合は、文字列に変換します。初期値：“”（空文字列）。
richtext ^(2.0)	△	真偽値	リッチテキスト（6）を有効にします。Professional 版のみ有効なエントリです。値が true の場合、value エントリで指定した文字列を XML 形式のリッチテキストとして扱います。初期値：false。
charref	○	真偽値	テキスト文字列中に文字参照を利用します。初期値：false（richtext が有効な場合は true）。
format	○	文字列	数値の書式を指定します。richtext が false かつ value 属性の値が数値として解釈できる場合に限り有効となります。value 属性の値が文字列の場合は、数値への変換を試みます。初期値：書式変換を行わない。
datetime	○	文字列	日付／時刻の書式を指定します。richtext が false かつ value 属性の値が日付もしくは時刻として解釈できる場合に限り有効となります。初期値：書式変換を行わない。
replace	○	文字列リスト	正規表現による文字列置換を指定します。richtext が false の場合に有効となります。「正規表現」にマッチした部分文字列を「置換テンプレート」で置き換えます。置換テンプレートには、\1 や\2 を含めることができます。これらは正規表現中の対応するグループにマッチした部分文字列で置き換えられます。＼0 は、正規表現全体にマッチした部分文字列を表します。書式：[正規表現 , 置換テンプレート]。初期値：文字列置換を行わない。例：{"replace": [".+", "\0 様"]}

format, datetime, replace の優先順位

format, datetime, replace の指定は排他的です。同時に指定された場合は、format → datetime → replace の優先順位で一つのみ適用されます。

数値書式指定文字列

format 属性において、数値書式指定文字列のプレースホルダとして利用可能な文字は、表 5.30 のとおりです。

表 5.30: 数値書式指定文字列

書式指定文字	意味
0	ゼロプレースホルダ
#	桁プレースホルダ
.	小数点
,	桁区切り記号、値の位取り
%	パーセントプレースホルダ
\ 文字	エスケープ文字
' 文字列 ' " 文字列 "	リテラル文字列
;	セクション区切り記号（最大 3 セクション：正；負；ゼロ）
その他の文字	結果の文字列にコピーされます。

数値書式指定文字列で利用可能なエスケープ文字は、表 5.31 のとおりです。

表 5.31: 書式指定文字列で利用可能なエスケープ文字

エスケープ文字	意味
\b	バックスペース (BS)
\n	行送り (LF)
\r	復帰 (CR)
\t	水平タブ (TAB)
\u HHHH	16 進数値 HHHH を持つ Unicode 文字
\ 文字	文字自身

数値書式指定文字列の使用例を表 5.32 に示します。

表 5.32: 数値書式指定文字列の使用例

書式指定文字列	フィールド値	外観文字列
####	123	123
0	123	123
(###)###-###	1234567890	(123)456-7890
#.##	1.2	1.2
0.00	1.2	1.20
00.00	1.2	01.20
,#	1234567890	1,234,567,890
,,	1234567890	1235
,,,	1234567890	1
,##0,,	1234567890	1,235
[##-##-##]	123456	[12-34-56]
##;(##)	1234	1234
##;(##)	-1234	(1234)

日付/時刻書式指定文字列

`datetime` 属性において、日付/時刻書式指定文字列のプレースホルダとして利用可能な文字は表 5.33 に示すとおりです。

表 5.33: 日付/時刻書式指定文字列

書式指定文字	意味
YY	2桁年（0パディングする）
YYYY	4桁年（0パディングする）
M	月（0パディングしない）
MM	月（0パディングする）
B	月略名（Jan., Feb., ... Dec.）
BB	月正式名（1月, 2月, ... 12月）
D	日（0パディングしない）
DD	日（0パディングする）
A	曜日略名（日, 月, ... 土）
AA	曜日正式名（日曜日, 月曜日, ... 土曜日）
G	年号略名（A.D., M, T, S, H）
GG	年号正式名（西暦, 明治, 大正, 昭和, 平成, 令和）
E	和暦（0パディングしない）
EE	和暦（0パディングする）
h	時（24時間表記, 0パディングしない）
hh	時（24時間表記, 0パディングする）
H	時（12時間表記, 0パディングしない）
HH	時（12時間表記, 0パディングする）
m	分（0パディングしない）
mm	分（0パディングする）
s	秒（0パディングしない）
ss	秒（0パディングする）
t	午前/午後略名（AM, PM）
tt	午前/午後正式名（午前, 午後）
\ 文字	エスケープ文字。数値書式指定文字のエスケープ文字と同じ。
‘ 文字列 ’ “ 文字列 ”	リテラル文字列
;	セクション区切り記号（最大3セクション：正；負；ゼロ）
他の文字	結果の文字列にコピーされます。

日付/時刻書式指定文字列の使用例を表 5.34 に示します。

表 5.34: 日付/時刻書式指定文字列の使用例

書式指定文字列	フィールド値	外観文字列
YYYY 年 MM 月 DD 曜日	2010-10-23T15:21:10	2010 年 10 月 23 日
GGEE 年 MM 月 DD 曜日	1911-04-04	明治 44 年 04 月 04 日
AA	2040-01-01	日曜日

正規表現

replace 属性において、文字列置換で利用できる正規表現は、表 5.35 のとおりです。

バックスラッシュ (\) を特殊文字として扱うプログラミング言語で使用する場合は、バックスラッシュ自身をエスケープする必要があります。

正規表現による文字列置換は、マルチバイト文字に対応していません。UTF-8において2バイト以上で表される文字は、1文字として扱われませんのでご注意ください。

表 5.35: 正規表現

特殊文字	説明
.	改行を除くすべての文字にマッチします。
*	(後置) 先行する正規表現の0回以上の繰り返しにマッチします。
+	(後置) 先行する正規表現の1回以上の繰り返しにマッチします。
?	(後置) 先行する正規表現の0回か1回の出現にマッチします。
[...]	文字集合。[a-z]のように-で範囲を表します。[^0-9]のように先頭に^を書くと補集合を取ります。]を含みたい場合には]を最初に書きます。-を含みたい場合には最初か最後に書きます。
^	行頭にマッチします(マッチさせる文字の先頭か、改行文字の直後にマッチします)。
\$	行末にマッチします(マッチさせる文字の末尾か、改行文字の直前にマッチします)。
	(中置) ふたつの正規表現の選択です。
\(..\)	囲まれた正規表現をグループ化し、名前をつけます。
\1	\(...\)\1でマッチした最初のテキスト(\2は2番目の式で、同様に\9まであります)。
\b	語の境界にマッチします。
\	特殊文字をクオートします。“\$^.*+?[]”が特殊文字です。

文字参照

charref 属性が有効な場合、テキスト文字列中に含まれる文字参照を文字に置き換えます。

文字参照には、HTML4.0で定義されている文字実体参照（付録 B 参照）と数値実体参照があります。

数値実体参照は、以下の書式で文字を指定します。

&#dddd; または &#xhhh;

ここで“ddd”は10進数、“hhh”は16進数のUnicodeコードポイントであり、桁数は任意です。

グリフ参照

Field Reports 独自のグリフ参照形式により、フォントに内蔵されている字形（グリフ）を直接指定することができます。

グリフ参照には、CID/GID 参照とグリフ名参照があります。

使用するフォントが OpenType の CJK フォント（拡張子：*.otf）であれば、CID をキーとしてグリフを指定することができます。CID とは、CID フォントが内蔵するすべてのグリフを一意に識別するために、Adobe 社が策定した番号です。日本語 CID フォントの場合、Adobe-Japan1(<https://github.com/adobe-type-tools/Adobe-Japan1/>) 文字コレクションにもとづいています。

使用するフォントが TrueType フォント（拡張子：*.ttf または *.ttc）であれば、GID（グリフ ID）をキーとしてグリフを指定することができます。一般に、GID はフォント固有の番号体系となっています。

CID/GID 参照は、以下の書式で指定します。

&#dddd; または &#hhhh;

ここで、“dddd”は10進数数、“hhhh”は16進数のCIDまたはGIDを示します。桁数は任意です。

TrueTypeフォントにおいて「グリフ名」が定義されている場合は、グリフ名によりグリフを指定することができます。グリフ名参照は、以下の書式で指定します。

<グリフ名>;

グリフ名が定義されているかどうかは、AFDKO(<https://github.com/adobe-type-tools/afdko>)などのツールで、TrueTypeフォントのpostテーブルの内容をダンプすると確認することができます。

グリフ名の定義を確認できているフォントとして、IPAフォント(<https://moji.or.jp/ipafont/>)とIPAmj明朝フォント(<https://moji.or.jp/mojikiban/font/>)があります(弊社調べ)。

5.9.6 テキスト・スタイル属性

テキストのスタイルを指定します。

表 5.36: テキスト・スタイル属性

キー	拡張	型	値
font	-	リソース名	テキストを描画する際に使用するフォントを指定します。resources 辞書 (5.6) で定義したフォント・リソース名またはシステム定義フォント名 (表 5.37) を指定します。初期値：“/KozGo-Medium”
font-size	-	長さ	フォントサイズを指定します。multiline が false の時に 0 を指定すると、テキストが文字枠に収まるようにフォントサイズを自動で設定します。multiline が true の時に 0 を指定した場合は、フォントサイズを 10.5 ポイントとします。初期値：0.
color	-	色	テキストの塗りつぶし色を指定します。初期値：[0] (黒).
font-stretch (2.0)	○	比率	フォントの縦横比を指定します。1.0 より大きい数値を指定した場合、横長の文字 (平体) となります。初期値：1.0.
font-kerning (2.0)	○	真理値	文字間のカーニングを行うかどうかを指定します。初期値：false.
letter-spacing (2.0)	○	長さ	文字間のスペースを増減します。初期値：0.
word-spacing (2.0)	○	長さ	単語間のスペースを増減します。初期値：0.
text-stroke-color	○	色	テキストの縁取り色を指定します。透明色を指定した場合、テキストの縁取りは表示しません。color に透明色を指定し、text-stroke-color に不透明色を指定した場合は、縁取りのみ表示します。初期値：[] (透明).
text-stroke-width (2.0)	○	長さ	テキストの縁取りの線幅を指定します。初期値：1.
text-paint-order (2.0)	○	列挙値	テキストの塗りつぶし色と縁取り色の描画順序を指定します。 Fill 塗りつぶし色、縁取り色の順に描画します。 Stroke 縁取り色、塗りつぶし色の順に描画します。 初期値：Fill.

システム定義フォント

フォント名として、表 5.37 のシステム定義フォントを使用することができます。

表 5.37: システム定義フォント

フォント名	説明
/Times-Roman	Times Roman 体フォント
/Times-Bold	Times ボールド体フォント
/Times-Italic	Times イタリック体フォント
/Times-BoldItalic	Times ボールド・イタリック体フォント
/Helvetica	Helvetica フォント
/Helvetica-Bold	Helvetica ボールド体フォント
/Helvetica-Oblique	Helvetica 斜体フォント
/Helvetica-BoldOblique	Helvetica ボールド斜体フォント
/Courier	Courier フォント
/Courier-Bold	Courier ボールドフォント
/Courier-Oblique	Courier 斜体フォント
/Courier-BoldOblique	Courier ボールド斜体フォント
/Symbol	Symbol フォント
/ZapfDingbats	ZapfDingbats フォント
/KozMin-Regular	小塚明朝体フォント
/KozGo-Medium	小塚ゴシック体フォント

カーニング

`font-kerning` により、カーニングの有無を制御できます（図 5.5）。実際のカーニング量はフォントより取得しますので、カーニング情報を持ったフォントを使用する必要があります。

AWAY (カーニングなし)

AWAY (カーニングあり)

図 5.5: カーニングの効果

text-paint-order

`text-stroke-width` によりテキストに縁取りを付けることができますが、縁取りの線幅が太くなると、文字が潰れてしまう問題があります。塗りつぶし色と縁取り色の描画順序を変更することで、この問題を回避することができます（図 5.6）。



図 5.6: text-paint-order の効果

5.9.7 テキスト・レイアウト属性

テキストフィールドにおいて、テキストのレイアウトを指定します。

表 5.38: テキスト・レイアウト属性

キー	拡張	型	値
text-align	△	列挙値	<p>テキストの整列方法を指定します。</p> <p>Left 左寄せ</p> <p>Center 中央寄せ</p> <p>Right 右寄せ</p> <p>Justify 均等割付（拡張属性）</p> <p>初期値：Left.</p>
text-align-last (2.0)	○	列挙値	<p>text-align が Justify の場合に、最終行または強制改行前のブロックの最終行の行揃えの方法を指定します。</p> <p>Auto text-justify が Distribute の場合、均等割付けします。</p> <p>その他の値の場合、左寄せとします。</p> <p>Left 左寄せ</p> <p>Center 中央寄せ</p> <p>Right 右寄せ</p> <p>Justify 均等割付</p> <p>初期値：Auto.</p>
text-justify (2.0)	○	列挙値	<p>text-align が Justify の場合、均等割の形式を指定します。</p> <p>Auto 文字間・単語間のスペースを自動的に調整します。</p> <p>InterWord 単語間のスペースを調整して、均等割付けを行います。</p> <p>Distribute 文字種に関係なく、文字間隔を均等に調整します。</p> <p>Stretch フォントの縦横比を調整することで、均等割付けを行います。 いわゆる長体・平体の処理を行います。</p> <p>初期値：multiline 属性が true の場合 Auto, false の場合 Distribute.</p>
text-spacing (2.0)	○	列挙値のリスト	<p>文字間スペースを自動挿入する際のルールを指定します。</p> <p>Normal [SpaceStart, SpaceEnd, TrimAdjacent] と指定した場合と同等です。</p> <p>None 文字間スペースの挿入を行いません。</p> <p>SpaceStart 行頭の全角開き括弧を全角幅とします。</p> <p>TrimStart 行頭の全角開き括弧を半角幅とします。</p> <p>SpaceEnd 行末の全角閉じ括弧を全角幅とします。</p> <p>TrimEnd 行末の全角閉じ括弧を半角幅とします。</p> <p>SpaceAdjacent 約物間のスペースを折り畳みません。</p> <p>TrimAdjacent 約物間のスペースを折り畳みます。</p> <p>IdeographAlpha 表意文字（アルファベット等）と非表意文字（漢字等）との間に、1/4em 幅のスペースを挿入します。</p> <p>IdeographNumeric 数字と非表意文字（漢字等）との間に、1/4em 幅のスペースを挿入します。</p> <p>NoCompress 均等割付けの調整を行う際に「詰め」の処理を行いません。</p> <p>単項目を指定する場合は、列挙値を指定します。複数指定する場合は、列挙値のリストとします。初期値：Normal.</p>

(続く)

(続き)

キー	拡張	型	値
stretch-range ^(2.0)	○	数値列	text-justify が Stretch の場合に、フォント扁平率の下限と上限を 2 要素の数値リストとして指定します。初期値：[0.5, 2.0].
text-overhang ^(2.0)	○	列挙値	テキストがフィールドに収まらない場合の挙動を指定します。 Auto テキストの両端を均等にはみ出させます。 Start テキストの先頭をはみ出させます。 End テキストの末尾をはみ出させます。 Shrink 長体を掛けて、テキストが収まるように調整します。 Reduce フォントサイズを縮小させて、収まるように調整します。 初期値：Auto.
line-height	○	長さ	行の送り幅を指定します。縦組みの場合も行送り幅としてこの値を利用します。単位を省略した場合、font-size に対する倍率(em)と解釈します。初期値：1.2.
line-skip-limit ^(2.0)	○	長さ	行と行の間の空きの最低値を指定します。行と行がこれ以下に接近しないように、行の送り幅を調整します。初期値：2.
text-indent ^(2.0)	○	長さ	テキスト先頭行をインデントします。初期値：0.
vertical-align	○	列挙値	縦方向の揃え位置を指定します。下記列挙値が使用できます。 Top 上端揃え Middle 中央揃え Bottom 下端揃え 初期値：multiline が true の場合は Top, false の場合は Middle.

テキストの寄せ

text-align により、テキストの寄せを指定できます（図 5.7）。



図 5.7: text-align の例

また text-justify を用いて、均等割付時のスペースの調整方法を細かく制御することができます（図 5.8）。



図 5.8: text-justify の例

text-overhang

図 5.9 に、text-overhang の違いによる挙動の変化を一例として示します。

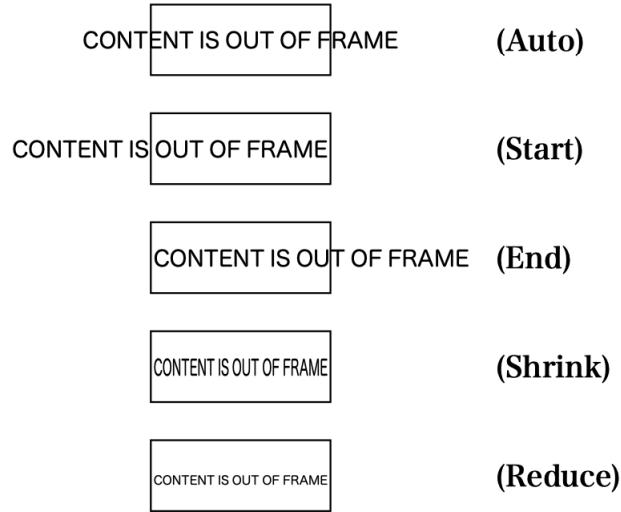


図 5.9: text-overhang の効果

5.9.8 テキスト改行属性

改行や空白文字の扱いを制御します。

表 5.39: テキスト改行属性

キー	拡張	型	値
multiline	-	真理値	テキストを複数行に分割することを指示します。初期値 : false.
word-break (2.0)	<input type="radio"/>	列挙値	<p>multiline が true の場合にテキストの折り返しを行う際のルールを指定します。</p> <p>Normal 言語の組版ルールに応じた箇所で自動的に折り返します。</p> <p>BreakAll 任意の 2 文字間で折り返します。</p> <p>KeepAll 単語間スペースの位置でのみ折り返します。</p> <p>初期値 : Normal.</p>
line-break (2.0)	<input type="radio"/>	列挙値	<p>word-break が Normal の場合に適用する禁則処理のルールを指定します。</p> <p>Normal 標準の禁則処理を行います。</p> <p>Strict 厳密な禁則処理を行います。</p> <p>初期値 : normal.</p>
hyphens	<input type="radio"/>	列挙値	<p>multiline が true の場合に、欧文単語のハイフネーション処理の挙動を指定します。</p> <p>Auto ハイフン挿入位置を自動決定します。ただし、ソフトハイフンが指定されている場合は、その位置を優先して使用します。</p> <p>Manual ソフトハイフンの処理のみ行います。</p> <p>None ハイフネーション処理を行いません。</p> <p>初期値 : Manual.</p>
white-space (2.0)	<input type="radio"/>	列挙値	<p>要素内のスペースの処理方法を設定します。</p> <p>Normal 連続する空白文字をひとつの空白にまとめます。必要な場合に、自動改行を行います。</p> <p>Nowrap 連続する空白文字をひとつの空白にまとめます。自動改行は行いません。</p> <p>Pre 空白文字を保持します。改行文字により行を折り返します。</p> <p>PreLine 連続する空白文字をひとつの空白にまとめます。改行文字により行を折り返します。必要な場合に、自動改行を行います。</p> <p>PreWrap 空白文字を保持します。改行文字により行を折り返します。必要な場合に、自動改行を行います。</p> <p>ただし、multiline が false の場合は、改行文字による折り返しも自動的に行いません。初期値 : richtext 属性が true の場合 Normal, false の場合 PreLine.</p>

5.9.9 ボタンフィールド属性

ボタン（画像）フィールドでは、表 5.40 に示す属性を設定することができます。

表 5.40: ボタン（画像）フィールド属性

キー	型	値
image	リソース名、 リソース名#ページ番号(2.0)	ボタン（画像）フィールドの「アイコン」として表示する 画像のリソース名を指定します。画像リソースは、image 属性辞書（5.18）で定義します。複数ページ画像リソース の特定ページを参照する際は、ページ番号（0 始まり）付 きのリソース名で指定します。
icon	URL	ボタン（画像）フィールドの「アイコン」として表示する 画像データの URL を指定します。
shape(2.0)	文字列	図形要素（7）を XML 文字列形式で記述します。shape 属性では、ルート要素 shape を省略して、任意数の図形要素 を記述します。
view-box(2.0)	長さリスト	（shape 指定時のみ）図形要素のユーザー座標系を定義 します。min-x と min-y で x 座標と y 座標の最小値を、 width と height で描画エリアの幅と高さを指定します。 書式 : [min-x, min-y, width, height].
text-align	列挙値	水平方向に余白がある場合の整列方法を指定します。 選択値 : Left, Center, Right. 初期値 : Center.
vertical-align	列挙値	垂直方向に余白がある場合の整列方法を指定します。 選択値 : Top, Middle, Bottom. 初期値 : Middle.

ボタンフィールドの外観として、image, icon, shape 属性のどれかを指定してください。何も指定しない場合は、空のフィールドとなります。

icon 属性で同じ URL が複数回出現する場合は、最初に取得したコンテンツが再利用されます。

ユーザー座標系

図形要素内では x が右方向、y が下方向に大きくなるユーザー座標系となります（図 5.10）。PDF 座標系とは、y 方向の向きが異なることに注意してください。view-box 省略時は、左上を原点 (0,0) とし、長さの単位は 1px=1pt となります。

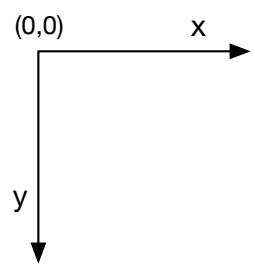


図 5.10: ユーザー座標系

5.9.10 可変テーブル属性

可変テーブル・フィールドでは、表 5.41 に示す属性を設定することができます。

表 5.41: 可変テーブル属性

キー	型	値
new または ftype (2.0)	列挙値	(必須) 固定値 : Tbl.
rect	長さリスト	(必須) フィールドの矩形座標を指定します。ページの左下を原点として、左下・右上座標を 4 要素のリストとして与えます。書式 : [<i>llx</i> , <i>lly</i> , <i>urx</i> , <i>ury</i>].
column-widths (2.0)	長さ・比率リスト	各列の幅をリスト形式で指定します。rect で指定したフィールド枠の幅から左右境界線の幅を引いた長さを各列に分配します。比率で指定する場合は、合計が 100% となるように按分してください。column-widths を指定した場合、このリストの要素数に基づきテーブルの列数を決定します。省略した場合は、body 属性の要素数に基づき列数を決定し、列幅は均等割となります。
row-height (2.0)	長さ	各行の高さを指定します。省略した場合、セルの内容物により高さが決定されます。
cell-padding (2.0)	長さ、長さリスト	セル余白の量を指定します。書式は、padding (5.9.3) と同様です。
body (2.0)	field 要素行列	(必須) ボディー部を構成するセルの値を 2 階層のリストで指定します。
header (2.0)	field 要素行列	ヘッダー部を構成するセルの値を 2 階層のリストで指定します。
footer (2.0)	field 要素行列	フッター部を構成するセルの値を 2 階層のリストで指定します。
border-columns (2.0)	境界線スタイル辞書	行と行の間の境界線スタイルを境界線スタイル辞書 (表 5.28) で指定します。
border-rows (2.0)	境界線スタイル辞書	列と列の間の境界線スタイルを境界線スタイル辞書 (表 5.28) で指定します。
border-header (2.0)	境界線スタイル辞書	ヘッダー部とボディー部の間の境界線スタイルを境界線スタイル辞書 (表 5.28) で指定します。
border-footer (2.0)	境界線スタイル辞書	ボディー部とフッター部の間の境界線スタイルを境界線スタイル辞書 (表 5.28) で指定します。
text-align	列挙値	水平方向に余白がある場合の整列方法を指定します。 選択値 : Left, Center, Right. 初期値 : Center.
vertical-align	列挙値	垂直方向に余白がある場合の整列方法を指定します。 選択値 : Top, Middle, Bottom. 初期値 : Top.

PDF 仕様には、可変テーブルに該当するフィールドが存在しないため、テンプレートに配置したフィールドを可変テーブルのプレースホルダとして利用することはできません。new 属性に Tbl を設定し、rect 属性に座標を指定してください。

5.10 property 辞書

PDF 帳票に設定するプロパティを表 5.42 の property 辞書により与えます。PDF テンプレートが元々持っているプロパティは、生成される PDF 帳票には引き継がれませんので、この property 辞書で明示的に設定する必要があります。

表 5.42: property 辞書のエントリ

キー	型	値
docinfo	辞書	「文書のプロパティ」として設定する値を指定します。省略時：文書のプロパティの設定を行わない。
metadata	URL	PDF に埋め込む metadata の URL を指定します。省略時：metadata の埋込みを行わない。
encryption	辞書	セキュリティの設定を行う場合に、暗号化パラメータを設定します。省略時：セキュリティの設定を行わない。
linearized	真偽値	「Web 表示用に最適化」を行うかどうかを指定します。初期値：False.
viewer-preferences	辞書	PDF をビューア・アプリケーションで開いた際の「開きかた」を指示します。省略時：「開きかた」の指示を行わない。

5.10.1 docinfo 辞書

docinfo 辞書では、Adobe Reader の「文書プロパティ」で、主に「概要」として表示される情報を設定します。表 5.43 に docinfo 辞書のエントリを示します。

表 5.43: docinfo 辞書のエントリ

キー	型	値
title	文字列	PDF の「文書情報辞書」の「Title」エントリの値を設定します。Adobe Reader の「文書のプロパティ」では「タイトル」として表示されます。
author	文字列	PDF の「文書情報辞書」の「Author」エントリの値を設定します。「作成者」として表示されます。
subject	文字列	PDF の「文書情報辞書」の「Subject」エントリの値を設定します。「サブタイトル」として表示されます。
keywords	文字列	PDF の「文書情報辞書」の「Keywords」エントリの値を設定します。「キーワード」として表示されます。
creator	文字列	PDF の「文書情報辞書」の「Creator」エントリの値を設定します。「アプリケーション」として表示されます。
producer	文字列	PDF の「文書情報辞書」の「Producer」エントリの値を設定します。初期値：“Field Reports”。
creation-date	文字列	PDF の「文書情報辞書」の「CreationDate」エントリの値を設定します。「作成日」として表示されます。初期値：PDF 帳票を生成した時刻。
mod-date	文字列	PDF の「文書情報辞書」の「ModDate」エントリの値を設定します。「作成日」として表示されます。初期値：PDF 帳票を生成した時刻。

5.10.2 metadata

docinfo に加えて XML 形式の詳細な文書情報を埋め込む際に使用します。

Adobe 社の定義した Extensible Metadata Platform (XMP) 規格 (<https://www.adobe.com/devnet/xmp.html>) に準拠した XML の URL を与えます。

5.10.3 encryption 辞書

encryption 辞書では、セキュリティの設定を指定します。表 5.44 に encryption 辞書のエントリを示します。

表 5.44: encryption 辞書のエントリ

キー	型	値
method	列挙値	(必須) 暗号アルゴリズムを指定します。選択値：RC4, AES
key-length	整数	暗号化キーの長さ (ビット数) を指定します。暗号アルゴリズムに AES を選択した場合は、この項目は無視され、常に 128 が選択されます。選択値：40, 128 初期値：128.
owner-password	文字列	オーナーパスワードを指定します。Adobe Acrobat では、「権限パスワード」と呼ばれます。owner-password または user-password のうちどちらかに 1 文字以上のパスワードを設定する必要があります。初期値：" " (空文字列).
user-password	文字列	ユーザーパスワードを指定します。Adobe Acrobat では、「文書を開くパスワード」と呼ばれます。owner-password または user-password のうちどちらかに 1 文字以上のパスワードを設定する必要があります。初期値：" " (空文字列).
permissions	整数または列挙値のリスト	文書がユーザーパスワードで開かれるときに許可すべきアクセス権限の種類を指定するフラグのセットを設定します。整数で指定する場合、各アクセス権限は 32 ビット整数のビット位置に対応します。初期値：0xFFFF (すべて許可).

ユーザーパスワードによるアクセス権限

permissions エントリにおいて指定可能なアクセス制限は、表 5.45 のとおりです。

表 5.45: ユーザーパスワードによるアクセス権限

ビット位置	列挙値	意味
1-2	-	予約：必ず 0 でなければなりません。
3	Print	高解像度での印刷
4	Edit	(テキスト注釈および対話フォームフィールド以外の) 文書内容の変更
5	Copy	文書からのテキストとグラフィックスのコピー
6	Annot	テキスト注釈および対話フォームフィールドの追加または変更
7-8	-	予約：必ず 1 でなければなりません。
9	Forms	対話フォームフィールドへの値の入力
10	Extract	スクリーンリーダーデバイスのテキストアクセスを有効にする。
11	Assemble	ページの挿入・削除・回転
12	HqPrint	低解像度での印刷
13-32	-	予約：必ず 1 でなければなりません。

5.10.4 viewer-preferences 辞書

表 5.46 に示す viewer-preferences 辞書では、PDF をビューア・アプリケーションで開いた時の開きかたについての指示を設定します。

表 5.46: viewer-preferences 辞書のエントリ

キー	型	値
hide-toolbar	真偽値	ビューア・アプリケーションのツールバーを隠すかどうかを指定します。初期値 : false.
hide-menubar	真偽値	ビューア・アプリケーションのメニューバーを隠すかどうかを指定します。初期値 : false.
hide-window-ui	真偽値	文書ウィンドウのユーザーインターフェース要素を隠すかどうかを指定します。初期値 : false.
fit-window	真偽値	最初に表示されるページのサイズに適合するように文書ウィンドウのサイズを変更するかどうかを指定します。初期値 : false.
center-window	真偽値	文書ウィンドウを画面の中央に配置するかどうかを指定します。初期値 : false.
display-doc-title	真偽値	文書ウィンドウのタイトルバーに文書プロパティの title を表示するかどうかを指定します。初期値 : false.
non-full-screen-page-mode	列挙値	フルスクリーンモードでない時の文書のページモードを指定します。選択値 : UseNone, UseOutlines, UseThumbs。初期値 : UseNone.
direction	列挙値	文書を読む方向を指定します。選択値 : L2R (左から右), R2L (右から左)。初期値 : L2R.
view-area	列挙値	ページの表示領域となる「境界」を指定します。選択値: MediaBox, CropBox, BleedBox, TrimBox, ArtBox. 初期値 : CropBox.
view-clip	列挙値	画面表示時のクリッピング領域となる「境界」を指定します。選択値 : MediaBox, CropBox, BleedBox, TrimBox, ArtBox. 初期値 : CropBox.
print-area	列挙値	印刷領域となる「境界」を指定します。選択値 : MediaBox, CropBox, BleedBox, TrimBox, ArtBox. 初期値 : CropBox.
print-clip	列挙値	印刷時のクリッピング領域となる「境界」を指定します。選択値 : MediaBox, CropBox, BleedBox, TrimBox, ArtBox. 初期値 : CropBox.
print-scaling	列挙値	文書を印刷する際に表示されるプリントダイアログの「印刷倍率」を指定します。選択値 : None (印刷倍率を変更しない), AppDefault (現在の印刷倍率を使用する). 初期値 : AppDefault.

5.11 環境変数

5.11.1 ユーザー定義環境変数

表 5.47 に示す environ 辞書により、ユーザー定義の環境変数を設定します。

表 5.47: environ 辞書のエントリ

キー名	型	値
環境変数名	文字列または数値	環境変数を定義し、任意の値を与えます。

5.11.2 システム定義環境変数

システム定義されている環境変数の一覧を表 5.48 に示します。

表 5.48: 既定の環境変数

変数名	値
PAGE	現在のページ数（0はじまり）
PAGE+	現在のページ数（1はじまり）
NUM_PAGES	全ページ数
NOW	現在時刻

PAGE, NUM_PAGES を使って、帳票全体を通したページ番号を取得できます。Field Reports 内部では、ページ番号を 0 始まりの数値で管理しているので、1 始まりのページ番号を取得したい場合は、PAGE+ を使用してください。

5.12 settings 辞書

Field Reports の設定情報を表 5.49 の settings 辞書により与えます。

表 5.49: settings 辞書のエントリ

キー	型	値
template-root	文字列	PDF テンプレート・画像・フォント・metadata ファイルが格納されているディレクトリを設定します。初期値：“”。
serial-number	文字列	製品ご購入時に発行されたシリアル番号を設定します。
auth-code	文字列	ライセンス認証手続きにより発行されたライセンスキーを設定します。

第 6 章

リッチテキスト (Professional 版)

Field Reports では、PDF 1.5 より追加された “Rich Text Strings” をベースとして独自の XML 文書を定義しています。

6.1 XML 要素

利用可能な要素の一覧を表 6.1 に示します。表中、拡張欄が○の項目は Field Reports で独自に拡張した要素を示します。

表 6.1: 利用可能な要素

要素名	拡張	説明
<body>	-	XML 文書のルート要素です。 ルート要素です。
<p>	-	段落として認識されるテキストを囲みます。 ブロックレベル要素です。
	-	スタイルを適する範囲を囲むために使用します。 インライン要素です。
<i>	-	PDF 仕様との互換性確保のための要素です。現状 span 要素と同じ働きをします。 インライン要素です。
	-	PDF 仕様との互換性確保のための要素です。現状 span 要素と同じ働きをします。 インライン要素です。
 	○	テキストをこの位置で改行します。
<ruby>	○	rt と合わせて対象のテキストにルビをふるために使用します。 インライン要素です。
<rt>	○	ルビ（読みがな）のテキストを指定します。 ruby 要素の配下に配置します。
	○	画像を配置します。 インライン要素です。
<shape>	○	図形を配置します。 インライン要素です。

6.1.1 文書構造

リッチテキストは、body をルート要素とした XML 形式の文字列として記述します。

```
<?xml version="1.0"?>
<body xmlns="http://www.w3.org/1999/xhtml" xmlns:xfa="http://www.xfa.org/schema/xfa-data/1.0/">
  <p> (段落 1) </p>
  <p> (段落 2) </p>
  ...
</body>
```

XML 宣言と名前空間の指定は省略できます。

```
<body>
  <p> (段落 1) </p>
  <p> (段落 2) </p>
  ...
</body>
```

- ルート要素の配下には、複数のブロックレベル要素を配置できます。
- ブロックレベル要素の配下には、複数のインライン要素を配置できます。
ただし、ブロックレベル要素の配下にブロックレベル要素を配置することはできません。
- インライン要素の配下には、複数のインライン要素を配置できます。

6.1.2 body 要素

リッチテキストのルート要素です。

属性	説明
style	要素のスタイルを指定します。

body 要素の幅・高さは、フィールドの幅・高さにより規定されます。

6.1.3 p 要素

段落として認識されるテキストを囲みます。

属性	説明
width	要素の幅を指定します。
height	要素の高さを指定します。
style	要素のスタイルを指定します。

横組の場合、width の省略値は body 要素の幅により、height の省略値は内容物により規定されます。width, height 属性では、内容領域にパディングと境界線を含んだ幅・高さを指定します。

```
<body>
  <p>吾輩は猫である。</p>
  <p>名前はまだ無い。</p>
</body>
```

6.1.4 span 要素

属性	説明
style	要素のスタイルを指定します。

一部のスタイルを変更したい場合に使用します。

```
<p>吾輩は<span style='color:red;font-size:16;'>猫</span>である。</p>
```

6.1.5 br 要素

改行位置を明示的に指定したい場合に使用します。

```
<p>吾輩は<br/>猫である。</p>
```

6.1.6 ruby 要素

属性	説明
style	ruby 要素のスタイルを指定します。

以下の書式で、親文字とルビの組を指定します。

```
<ruby>愛知県<rt>あいちけん</rt></ruby>
```

モノルビを指定する場合は、ruby 要素の中に親文字と rt 要素の組を複数指定します。

```
<ruby>愛<rt>あい</rt>知<rt>ち</rt>県<rt>けん</rt></ruby>
```

あいちけん
愛知県

あいちけん
愛知県

Aichi Pref.

ruby 要素は、入れ子にすることができます。

```
<ruby>
  <ruby style='ruby-position:after;'>
    愛知県
    <rt>あいちけん</rt>
  </ruby>
  <rt>Aichi Pref.</rt>
</ruby>
```

Aichi Pref.
愛知県
あいちけん

6.1.7 rt 要素

属性	説明
style	rt 要素のスタイルを指定します。

<ruby>～</ruby>の中で、子要素として使用します。

6.1.8 img 要素

属性	説明
image	画像リソース名を指定します。
src	画像 URL を指定します。
width	画像の幅を指定します。
height	画像の高さを指定します。
style	要素のスタイルを指定します。

以下の書式で、resources 辞書 (5.6) で定義した画像リソース名を指定します。

```
<img image='画像リソース名' width='100' />
```

- image または src 属性のどちらかの属性は必須です。
- width と height の両方を指定した場合、画像自体の縦横比は無視されます。
- width と height の両方を省略した場合、画像の解像度を元に計算したサイズを用います。
- style では、ボックス・スタイル属性（表 6.2）、境界線スタイル属性（表 6.3）ならびに text-align、vertical-align が有効です。

6.1.9 shape 要素

属性	説明
width	(必須) 図形要素の幅を指定します。
height	(必須) 図形要素の高さを指定します。
viewBox	図形要素の座標系を定義します。
style	要素のスタイルを指定します。

<shape>～</shape>の配下に任意個数の図形要素（第 7 章参照）を配置します。

```
<shape width='80mm' height='80mm' viewBox='0 0 100 100'>
  <line x1='5' y1='5' x2='90' y2='90' style='stroke:fuchsia; stroke-width:4;'/>
  <line x1='5' y1='10' x2='5' y2='90' style='stroke:red; stroke-width:2;'/>
</shape>
```

- style では、ボックス・スタイル属性（表 6.2）、境界線スタイル属性（表 6.3）ならびに text-align, vertical-align が有効です。

6.2 属性

6.2.1 style 属性

各要素で利用可能なスタイルの一覧を以下に示します。

フィールド属性（5.9）と同じ名称の style 属性は、同様の働きをします。フィールド属性で設定した値は、同名称の style 属性の初期値となります。

ボックス・スタイル属性

表 6.2: ボックス・スタイル属性

スタイル名	型	継承	説明
padding	長さ	×	境界線と内容物の間の余白の量を指定します。左下右上のパディングをまとめて指定する場合は、数値を1つ指定します。個別に指定する場合は、スペースで区切って複数の値を指定します。値を2つ指定した場合：左右と上下のパディングを指定。値を4つ指定した場合：左、下、右、上のパディングをそれぞれ指定。初期値：0.
padding-left	長さ	×	左側境界線と内容物の間の余白の量を指定します。
padding-bottom	長さ	×	下側境界線と内容物の間の余白の量を指定します。
padding-right	長さ	×	右側境界線と内容物の間の余白の量を指定します。
padding-top	長さ	×	上側境界線と内容物の間の余白の量を指定します。
background-color	色	×	要素の背景色を指定します。初期値：透明色.

境界線スタイル属性

表 6.3: 境界線スタイル属性

スタイル名	型	継承	説明
border-width	長さ	×	境界線の太さを指定します。0を指定すると、境界線が表示されません。初期値：0.
border-left-width	長さ	×	左境界線の太さを指定します。
border-bottom-width	長さ	×	下境界線の太さを指定します。
border-right-width	長さ	×	右境界線の太さを指定します。
border-top-width	長さ	×	上境界線の太さを指定します。
border-color	色	×	境界線の表示色を指定します。初期値：black.
border-left-color	色	×	左境界線の表示色を指定します。
border-bottom-color	色	×	下境界線の表示色を指定します。
border-right-color	色	×	左境界線の表示色を指定します。
border-top-color	色	×	上境界線の表示色を指定します。
border-style	列挙値	×	境界線のスタイルを指定します。 選択値：solid, dashed, beveled, inset, underline. 初期値：solid.
border-left-style	列挙値	×	左境界線のスタイルを指定します。
border-bottom-style	列挙値	×	下境界線のスタイルを指定します。
border-right-style	列挙値	×	右境界線のスタイルを指定します。
border-top-style	列挙値	×	上境界線のスタイルを指定します。
border-dash	長さリスト	×	境界線の破線パターンを指定します。
border-left-dash	長さリスト	×	左境界線の破線パターンを指定します。
border-bottom-dash	長さリスト	×	下境界線の破線パターンを指定します。
border-right-dash	長さリスト	×	右境界線の破線パターンを指定します。
border-top-dash	長さリスト	×	上境界線の破線パターンを指定します。
border-join-style	列挙値	×	境界線のスタイルを指定します。 miter-join 境界線の角を直角に描きます。 round-join 境界線の角を丸めます。 bevel-join 境界線の角を切り落とします。 初期値：miter-join.
border-radius (2.0)	長さリスト	×	境界線の角の丸めを指定します。1要素の場合、角の丸めに用いる円の半径を指定します。2要素の場合、角の丸めに用いる楕円のX軸半径とY軸半径を指定します。border-styleがSolidまたはDashedであり、かつ上下左右の境界線スタイルが等しい場合にのみ有効です。初期値："0 0".

テキスト・スタイル属性

表 6.4: テキスト・スタイル属性

スタイル名	型	継承	説明
font	リソース名	○	テキストを描画する際に使用するフォントを指定します。resources 辞書 (5.6) で定義したフォント・リソース名またはシステム定義フォント名 (5.37) を指定します。
font-size	長さ, 比率	○	フォントサイズを指定します。比率で指定した場合は、現在のフォントサイズを基準とします。初期値：10.5.
color	色	○	テキストの塗りつぶし色を指定します。初期値：black.
font-stretch	比率	○	フォントの縦横比を指定します。初期値：1.0.
font-kerning	列挙値	○	文字間のカーニングを行うかどうかを指定します。 auto, manual 文字間カーニングを適用します。 none 文字間カーニングを適用しません。 初期値：auto.
letter-spacing	長さ	○	文字間のスペースを増減します。初期値：0.
word-spacing	長さ	○	単語間のスペースを増減します。初期値：0.
text-stroke-color	色	○	テキストの縁取り色を指定します。透明色を指定した場合、テキストの縁取りは表示しません。color に透明色を指定し、text-stroke-color に不透明色を指定した場合は、縁取りのみ表示します。初期値：透明色.
text-stroke-width	長さ	○	テキストの縁取りの線幅を指定します。初期値：1.

テキスト・レイアウト属性

ブロックレベル要素で有効な属性です。

表 6.5: テキスト・レイアウト属性

スタイル名	型	継承	説明
text-align	列挙値	○	行揃えの方法や均等割付の指定を行います。 left 左寄せ center 中央寄せ right 右寄せ justify 均等割付 初期値 : left.
text-align-last	列挙値	○	text-align が justify の場合に、最終行または強制改行前のブロックの最終行の行揃えの方法を指定します。 auto text-justify が distribute の場合、均等割付けします。text-justify がその他の値の場合は、左寄せとします。 left 左寄せ center 中央寄せ right 右寄せ justify 均等割付 初期値 : auto.
text-justify	列挙値	○	text-align が justify の場合、均等割の形式を指定します。 auto 文字間・単語間のスペースを自動的に調整します。 inter-word 単語間のスペースを調整して、均等割付けを行います。 distribute 文字種に関係なく、文字間隔を均等に調整します。 stretch フォントの縦横比を調整することで、均等割付けを行います。いわゆる長体・平体の処理を行います。 初期値 : auto.
text-spacing	列挙値	○	文字間スペースを自動挿入する際のルールを指定します。 normal “space-start space-end trim-adjacent” と同等です。 none 文字間スペースの挿入を行いません。 space-start 行頭の全角開き括弧を全角幅とします。 trim-start 行頭の全角開き括弧を半角幅とします。 space-end 行末の全角閉じ括弧を全角幅とします。 trim-end 行末の全角閉じ括弧を半角幅とします。 space-adjacent 約物間のスペースを折り畳みませ n。 trim-adjacent 約物間のスペースを折り畳みます。 ideograph-alpha 表意文字（アルファベット等）と非表意文字（漢字等）との間に 1/4em 幅のスペースを挿入します。 ideograph-numeric 数字と非表意文字（漢字等）との間に 1/4em 幅のスペースを挿入します。 no-compress 均等割付けの調整を行う際に「詰め」の処理を行いません。 初期値 : normal.

(続く)

(続き)

スタイル名	型	継承	説明
stretch-range	数値列	○	フォント扁平率の下限と上限をスペース区切りの数値で指定します。text-justify で stretch を選択した際に参照されます。初期値：“0.5 2.0”。
line-height	長さ	○	行の高さを指定します。初期値：1.2.
text-indent	長さ	×	p 要素のテキスト先頭行をインデントします。初期値：0.
vertical-align	列挙値、長さ、比率	×	要素の縦方向の揃え位置を指定します。 baseline 適用した要素のベースラインを親要素のベースラインに揃えます。 top 上端揃え middle 中央揃え bottom 下端揃え text-top 上端揃え（親要素の上端に揃えます） text-bottom 下端揃え（親要素の下端に揃えます） super 上付き sub 下付き 長さ ベースライン位置を基準として、正の値なら上へ移動します。 比率 line-height の値に対する割合を指定します。 初期値：baseline.

改行属性

表 6.6: 改行属性

スタイル名	型	継承	説明
text-wrap	列挙値	○	<p>テキストの幅がボックス要素の幅を超えた場合に、行を折り返して複数行にするかどうかを指定します。field 辞書（5.9.8）の multiline 要素に対応します。</p> <p>normal テキストの折り返しを行います。</p> <p>none テキストの折り返しを行いません。要素に収まらないテキストは、はみ出でて表示されます。</p> <p>初期値：field 辞書の multiline が true の場合は normal, false の場合は none.</p>
word-break	列挙値	○	<p>text-wrap が normal の場合にテキストの折り返しを行う際のルールを指定します。</p> <p>normal 言語の組版ルールに応じた箇所で自動的に折り返します。</p> <p>break-all 任意の 2 文字間で折り返します。</p> <p>keep-all 単語間スペースの位置でのみ折り返します。</p> <p>初期値：normal.</p>
line-break	列挙値	○	<p>word-break が normal の場合に適用する禁則処理のルールを指定します。</p> <p>normal 標準の禁則処理を行います。</p> <p>strict 厳密な禁則処理を行います。</p> <p>初期値：normal.</p>
hyphens	列挙値	○	<p>text-wrap が normal の場合に、欧文単語のハイフネーション処理の挙動を指定します。</p> <p>auto ハイフン挿入位置を自動決定します。ただし、ソフトハイフンが指定されている場合は、その位置を優先して使用します。</p> <p>manual ソフトハイフンの処理のみ行います。</p> <p>none ハイフネーション処理を行いません。</p> <p>初期値：manual.</p>
white-space	列挙値	○	<p>要素内のスペースの処理方法を設定します。</p> <p>normal 連続する空白文字をひとつの空白にまとめます。必要な場合、自動改行を行います。</p> <p>nowrap 連続する空白文字をひとつの空白にまとめます。自動改行を行いません。</p> <p>pre 空白文字を保持します。改行文字により行を折り返します。</p> <p>pre-line 連続する空白文字をひとつの空白にまとめます。改行文字により行を折り返します。必要な場合は、自動改行を行います。</p> <p>pre-wrap 空白文字を保持します。改行文字により行を折り返します。必要な場合は、自動改行を行います。</p> <p>ただし、text-wrap が none の場合は、改行文字による行の折り返し、自動的な行の折り返しは行いません。初期値：normal.</p>

ruby 要素で利用可能なスタイル属性

ruby 要素では、通常のスタイル指定に加えて、表 6.7 に示すスタイルの指定が可能です。

表 6.7: ruby 要素で利用可能なスタイル属性

スタイル名	型	継承	説明
ruby-position	列挙値	○	ルビ文字の表示位置を指定します。 <code>before</code> 親文字の上（縦組みの場合右）にルビ文字を配置します。 <code>after</code> 親文字の下（縦組みの場合左）にルビ文字を配置します。 初期値 : <code>before</code> .
ruby-offset	長さ	○	ルビと親文字の間隔を指定します。負の値を指定した場合、ルビと親文字の間隔が縮まります。初期値 : 0.
ruby-align	列挙値	○	ルビの行揃えの位置や均等割付の指定を行います。 <code>left</code> 左寄せ <code>center</code> 中央寄せ <code>right</code> 右寄せ <code>distribute-letter</code> 均等割付 <code>distribute-space</code> ルビの前後にスペースを加えた上で、均等割付 初期値 : <code>distribute-space</code> .

6.2.2 長さの指定

長さを指定する場合、以下の単位が利用できます。単位を省略した場合は、特に注記のないかぎりポイント(pt) 単位となります。

`em` 一文字分の長さ (= フォントサイズ)

`ex` 英小文字「x」一文字分の高さ (= フォントサイズの 1/2)

`mm` ミリメートル

`cm` センチメートル

`in` インチ (1in = 2.54cm)

`pt` ポイント (1pt = 1/72in=0.3528mm)

`pc` パイカ (1pc = 12pt)

`q` 級 (1q = 0.709pt)

6.2.3 比率の指定

比率を指定する場合、単位記号 “%” を付けて百分率（パーセント）で指定するか、0 から 1 の数値で指定します (1.0 = 100%)。

6.2.4 色の指定

数値による指定

色を指定する場合、以下の書式で行います。

RGB 色 → # 6 桁 16 進数値

色名 → 列挙値

RGB 色 → **rgb(r,g,b)**

CMYK 色 → **cmyk(c,m,y,k)**

rgb(r,g,b) 形式の場合、色成分値を数値 (0~255) または百分率で指定します。

cmky(c,m,y,k) 形式の場合、色成分値を数値 (0~1) または百分率で指定します。

色名による指定

色名として指定できる値の一覧は以下の通りです。

表 6.8: 色名の一覧

列挙値	RGB 値
transparent (透明)	-
black	0, 0, 0
gray	128, 128, 128
silver	192, 192, 192
white	255, 255, 255
maroon	128, 0, 0
red	255, 0, 0
purple	128, 0, 128
fuchsia	255, 0, 255
green	0, 128, 0
lime	0, 255, 0
olive	128, 128, 0
yellow	255, 255, 0
navy	0, 0, 128
blue	0, 0, 255
teal	0, 128, 128
aqua	0, 255, 255

第 7 章

図形要素

7.1 XML 要素

図形要素を XML 形式の文字列として記述します。

Field Reports の図形要素は、概ね SVG のサブセットとなっています。利用可能な要素の一覧を以下に示します。

表 7.1: 図形要素

要素名	説明
<shape>	XML 文書のルート要素です。
<line>	始点から終点をもとに、線分を定義します。
<rect>	始点と幅・高さをもとに、矩形を定義します。
<circle>	中心点と半径をもとに、円を定義します。
<ellipse>	中心点と 2 つの半径をもとに、楕円を定義します。
<polyline>	複数を線分をつなげた折れ線を定義します。
<polygon>	複数のつながった線分で構成される多角形を定義します。
<path>	一連の連結された線・楕円弧・曲線をもとに任意の図形の輪郭を定義します。
<g>	複数の図形要素をグループ化します。

7.1.1 記述方法

画像フィールドとして記述する場合は、shape 属性の値として、図形要素を列挙します。

```
{  
  "new": "Btn",  
  "rect": ["20mm", "110mm", "100mm", "190mm"],  
  "view-box": [0, 0, 120, 120],  
  "border-width": 1,  
  "shape": "<circle cx='30' cy='30' r='20' style='stroke:black; fill:none;'/> \  
           <circle cx='80' cy='30' r='20' style='stroke-width:5; stroke:black; fill:none;'/> \  
           <ellipse cx='30' cy='80' rx='10' ry='20' style='stroke:black; fill:none;'/> \  
           <ellipse cx='80' cy='80' rx='20' ry='10' style='stroke:black; fill:none;'/>"  
}
```

リッチテキストの一要素として記述する際は、shape 要素の配下に図形要素を列挙します。

```
{  
    "new": "Tx",  
    "font": "HiraMinPro-W6",  
    "font-size": 24,  
    "value": "<body><p>あい<shape width='6em' height='6em' viewBox='0 0 100 100'> \  
        <rect x='10' y='10' width='80' height='60' style='fill:blue;' /></shape>うえお</p></body>",  
    "text-align": "Left",  
    "vertical-align": "Top",  
    "border-width": 2,  
    "border-style": "Solid",  
    "border-color": "Blue",  
    "richtext": true,  
    "line-height": 1.8,  
    "multiline": true,  
    "clipping": false,  
    "rect": [100, 450, 400, 750]  
}
```

7.1.2 line 要素

属性	説明
x1	(必須) 直線の始点 x 座標を指定します。
y1	(必須) 直線の始点 y 座標を指定します。
x2	(必須) 直線の終点 x 座標を指定します。
y2	(必須) 直線の終点 y 座標を指定します。
transform	座標系変換を指定します。
style	直線のスタイルを指定します。

```
<line x1='5' y1='5' x2='90' y2='90' style='stroke:fuchsia; stroke-width:4;'/>
```

7.1.3 rect 要素

属性	説明
x	(必須) 矩形の左辺の x 座標を指定します。
y	(必須) 矩形の左辺の y 座標を指定します。
width	(必須) 矩形の幅を指定します。
height	(必須) 矩形の高さを指定します。
rx	矩形の角を丸めるために用いる楕円の x 軸半径の長さを指定します。
ry	矩形の角を丸めるために用いる楕円の y 軸半径の長さを指定します。
transform	座標系変換を指定します。
style	矩形のスタイルを指定します。

```
<rect x='10' y='70' width='25' height='30' style='fill:#0000ff; stroke:red; stroke-width:7;' />
```

7.1.4 circle 要素

属性	説明
cx	(必須) 円の中心の x 座標を指定します。
cy	(必須) 円の中心の y 座標を指定します。
r	(必須) 円の半径を指定します。
transform	座標系変換を指定します。
style	円のスタイルを指定します。

```
<circle cx='30' cy='30' r='20' style='stroke:black; fill:none;' />
```

7.1.5 ellipse 要素

属性	説明
cx	(必須) 楕円の中心の x 座標を指定します。
cy	(必須) 楕円の中心の y 座標を指定します。
rx	(必須) 楕円の x 軸方向の半径を指定します。
ry	(必須) 楕円の y 軸方向の半径を指定します。
transform	座標系変換を指定します。
style	楕円のスタイルを指定します。

```
<ellipse cx='30' cy='80' rx='10' ry='20' style='stroke:black; fill:none;' />
```

7.1.6 polyline 要素

属性	説明
points	(必須) 折れ線を構成する一連の点を指定します。
transform	座標系変換を指定します。
style	折れ線のスタイルを指定します。

```
<polyline style='stroke:black; stroke-width:3; fill:none;'  
points='5 20 20 20 25 10 35 30 45 10 55 30 65 10 75 30 80 20 95 20' />
```

7.1.7 polygon 要素

属性	説明
points	(必須) 多角形を構成する一連の点を指定します。
transform	座標系変換を指定します。
style	多角形のスタイルを指定します。

```
<polygon points='15,10 55,10 45,20 5,20' style='fill:red; stroke:black;' />
```

7.1.8 path 要素

属性	説明
d	(必須) パスにより図形の外形線を指定します。
transform	座標系変換を指定します。
style	パスのスタイルを指定します。

```
<path d='M50,50 Q70,200 150,100 T270,70'  
      style='fill:none; stroke:#CF3721; stroke-width:3;'/>
```

7.1.9 g 要素

属性	説明
transform	座標系変換を指定します。
style	グループのスタイルを指定します。

```
<g style='stroke:black; fill:none;'>  
  <path d='M 10 10 L 100 10' />  
  <path d='M 10,20 L 100,20 L 100,50' />  
  <path d='M40 60 L 10,60 L 40 42.68 M60,60 L 90 60 L 60,42.68' />  
</g>
```

7.2 属性

7.2.1 transform 属性

transform 属性は、要素とその要素の子に適用される変換定義のリストを定義します。

表 7.2: 変換

変換名	説明
translate(x y)	座標系を移動させます。
scale(sx [sy])	座標系を拡大縮小します。
rotate(a [x y])	指定された点 (x, y) を軸に a 度回転させます。
skewX(a)	x 軸を基準に a 度傾斜変換させます。
skewY(a)	y 軸を基準に a 度傾斜変換させます。
matrix(a b c d e f)	6 つの値の変換行列の形式で変形を指定します。

7.2.2 style 属性

各要素で利用可能なスタイルの一覧を以下に示します。

表 7.3: style 属性

スタイル名	型	説明
stroke	色	枠線の描画色を指定します。初期値：none.
stroke-width	長さ	枠線の幅を指定します。初期値：1.0.
stroke-dasharray	長さリスト	枠線の描画に用いられる破線の間隔とパターンを指定します。破線と間隔の長さを交互に指定します。
stroke-dashoffset	長さ	破線パターンを描画する際にずらす距離を指定します。
stroke-linecap	列挙値	枠線の端点の形状を指定します。 選択値：butt, round, square. 初期値：butt.
stroke-linejoin	列挙値	枠線の角の部分の形状を指定します。 miter 境界線の角を直角に描きます。 round 境界線の角を丸めます。 bevel 境界線の角を切り落とします。 初期値：miter.
stroke-miterlimit	長さ	枠線のつなぎ目が鋭角になっている状況で、留め幅の stroke-width に対する限界比率を指定します。初期値：4.0.
fill	色	塗りつぶし色を指定します。初期値：black.
fill-rule	列挙値	図形の内側とされる領域を決定するためのアルゴリズムを指定します。 選択値：nonzero, evenodd. 初期値：nonzero.

7.2.3 長さの指定

長さを指定する場合、以下の単位が利用できます。単位を省略した場合は、ピクセル（px）単位となります。

px ピクセル（view-box の指定がない場合、1px=1pt）

mm ミリメートル

cm センチメートル

in インチ（1in = 2.54cm）

pt ポイント（1pt = 1/72in=0.3528mm）

pc パイカ（1pc = 12pt）

q 級（1q = 0.709pt）

7.2.4 色の指定

数値による指定

色を指定する場合、以下の書式で行います。

RGB 色 → # 6 桁 16 進数値

色名 → 列挙値

RGB 色 → **rgb(r,g,b)**

CMYK 色 → **cmyk(c,m,y,k)**

rgb(r,g,b) 形式の場合、色成分値を数値 (0~255) または百分率で指定します。

cmky(c,m,y,k) 形式の場合、色成分値を数値 (0~1) または百分率で指定します。

色名による指定

色名として指定できる値の一覧は以下の通りです。

表 7.4: 色名の一覧

列挙値	RGB 値
none (透明)	-
black	0, 0, 0
gray	128, 128, 128
silver	192, 192, 192
white	255, 255, 255
maroon	128, 0, 0
red	255, 0, 0
purple	128, 0, 128
fuchsia	255, 0, 255
green	0, 128, 0
lime	0, 255, 0
olive	128, 128, 0
yellow	255, 255, 0
navy	0, 0, 128
blue	0, 0, 255
teal	0, 128, 128
aqua	0, 255, 255

第 8 章

API リファレンス

Field Reports で利用可能な API の一覧を表 8.1 に示します。

表 8.1: API 一覧

分類	名称	提供形態
言語 Bridge	Python Bridge API (8.1)	Python 拡張モジュール
	Ruby Bridge API (8.2)	Ruby 拡張モジュール
	PHP Bridge API (8.3)	PHP 拡張モジュール
	Java VM Bridge API (8.4)	jar ファイル
	.NET Bridge API (8.5)	NuGet パッケージ
コマンドライン	コマンドライン I/F (8.6)	実行ファイル
サーバーモード	Web API (8.7)	〃

8.1 Python Bridge API

モジュール名 field_reports

8.1.1 Bridge クラス

Field Reports と連携するための Proxy オブジェクトを生成します。

`create_proxy(uri = None)`

URI に応じた Field Reports Proxy オブジェクトを返却します。

引数

`uri (str, optional)` Field Reports との接続方法を示す URI。

None を指定または省略した場合、環境変数 'REPORTS_PROXY' から URI を取得します。

環境変数 'REPORTS_PROXY' も未設定の場合の既定値は"exec:reports"です。

書式 (コマンド連携時)

`exec:{exePath}?cwd={cwd}&loglevel={logLevel}`

- cwd, loglevel は省略可能です。
- loglevel が 0 より大きい場合、標準エラー出力にログを出力します。

書式 (HTTP 連携時)

```
http://{hostName}:{portNumber}/
```

返り値

Proxy Field Reports Proxy オブジェクト

サンプルコード (コマンド連携時)

```
from field_reports import Bridge
reports = Bridge.create_proxy("exec:/usr/local/bin/reports?cwd=/usr/share&loglevel=3");
```

サンプルコード (HTTP 連携時)

```
from field_reports import Bridge
reports = Bridge.create_proxy("http://localhost:50080/");
```

`create_exec_proxy(exe_path="reports", cwd=". ", loglevel=0, logout=sys.stderr)`

コマンド呼び出しにより Field Reports と連携する Proxy オブジェクトを生成します。

引数

`exe_path` (str, optional) Field Reports コマンドのパス

`cwd` (str, optional) Field Reports プロセス実行時のカレントディレクトリ

`loglevel` (int, optional) ログ出力レベル (0: ログを出力しない, 1: ERROR ログ, 2: WARN ログ, 3: INFO ログ, 4: DEBUG ログ)

`logout` (TextIO, optional) ログ出力先 Stream

返り値

Proxy Field Reports Proxy オブジェクト

`create_http_proxy(base_address="http://localhost:50080/")`

HTTP 通信により Field Reports と連携する Proxy オブジェクトを生成します。

引数

`base_address` (str, optional) ベース URI

返り値

Proxy Field Reports Proxy オブジェクト

8.1.2 Proxy インターフェース

`version(self)`

バージョン番号を取得します。

返り値

`str` バージョン番号

例外

`ReportsError` Field Reports との連携に失敗した場合に発生

`render(self, param)`

レンダリング・パラメータを元にレンダリングを実行します。

引数

`param (str|bytes|dict)` JSON 文字列または辞書形式レンダリング・パラメータ ([5 参照](#))

返り値

`bytes` PDF データ

例外

`ReportsError` Field Reports との連携に失敗した場合に発生

`parse(self, pdf)`

PDF データを解析し、フィールドや注釈の情報を取得します。

引数

`pdf (bytes)` PDF データ

返り値

`dict` 解析結果

例外

`ReportsError` Field Reports との連携に失敗した場合に発生

8.1.3 ExeProxy クラス

コマンド呼び出しにより Field Reports との連携を行う、Proxy インターフェースの実装クラスです。

8.1.4 HttpProxy クラス

HTTP 通信により Field Reports との連携を行う、Proxy インターフェースの実装クラスです。

8.1.5 サンプルプログラム

```
# -*- coding: utf-8 -*-

from field_reports import Bridge, ReportsError

param = {
    "template": {"paper": "A4"},
    "context": {
        "hello": {
            "new": "Tx",
            "value": "Hello, World!",
            "rect": [100, 700, 400, 750]
        }
    }
}
reports = Bridge.create_proxy()
pdf = reports.render(param)
with open("out.pdf", "wb") as f:
    f.write(pdf)
```

コード 8.1: hello.py

8.2 Ruby Bridge API

モジュールファイル名 field_reports

モジュール名 FieldReports

8.2.1 FieldReports::Bridge クラス

Field Reports と連携するための Proxy オブジェクトを生成します。

`create_proxy(uri = nil)`

URI に応じた Field Reports Proxy オブジェクトを返却します。

引数

`uri (String)` Field Reports との接続方法を示す URI。

nil を指定または省略した場合、環境変数 ‘REPORTS_PROXY’ から URI を取得します。

環境変数 ‘REPORTS_PROXY’ も未設定の場合の既定値は “exec:reports” です。

書式 (コマンド連携時)

`exec:{exePath}? cwd={cwd}& loglevel={logLevel}`

- cwd, loglevel は省略可能です。
- loglevel が 0 より大きい場合、標準エラー出力にログを出力します。

書式 (HTTP 連携時)

```
http://{hostName}:{portNumber}/
```

返り値

Proxy Field Reports Proxy オブジェクト

サンプルコード (コマンド連携時)

```
require "field_reports"

reports = FieldReports::Bridge.create_proxy("exec:/usr/local/bin/reports?cwd=/usr/
share&loglevel=3")
```

サンプルコード (HTTP 連携時)

```
require "field_reports"

reports = FieldReports::Bridge.create_proxy("http://localhost:50080/")
```

`create_exec_proxy(exe_path="reports", cwd=". ", loglevel=0, logout=STDERR)`

コマンド呼び出しにより Field Reports と連携する Proxy オブジェクトを生成します。

引数

`exe_path (String)` Field Reports コマンドのパス

`cwd (String)` Field Reports プロセス実行時のカレントディレクトリ

`loglevel (Integer)` ログ出力レベル (0: ログを出力しない, 1: ERROR ログ, 2: WARN ログ, 3: INFO ログ, 4: DEBUG ログ)

`logout (IO)` ログ出力先 Stream

返り値

Proxy Field Reports Proxy オブジェクト

`create_http_proxy(base_address="http://localhost:50080/")`

HTTP 通信により Field Reports と連携する Proxy オブジェクトを生成します。

引数

`base_address (String)` ベース URI

返り値

Proxy Field Reports Proxy オブジェクト

8.2.2 FieldReports::Proxy インターフェース

`version`

バージョン番号を取得します。

返り値

`String` バージョン番号

例外

`ReportsError` Field Reports との連携に失敗した場合に発生

`render(param)`

レンダリング・パラメータを元にレンダリングを実行します。

引数

`param (String|Hash)` JSON 文字列またはハッシュ値レンダリング・パラメータ ([5 参照](#))

返り値

`String` PDF データ

例外

`ReportsError` Field Reports との連携に失敗した場合に発生

`parse(pdf)`

PDF データを解析し、フィールドや注釈の情報を取得します。

引数

`pdf (String)` PDF データ

返り値

`String` 解析結果

例外

`ReportsError` Field Reports との連携に失敗した場合に発生

8.2.3 ExeProxy クラス

コマンド呼び出しにより Field Reports との連携を行う、Proxy インターフェースの実装クラスです。

8.2.4 HttpProxy クラス

HTTP 通信により Field Reports との連携を行う、Proxy インターフェースの実装クラスです。

8.2.5 ReportsError クラス

Field Reports との連携に失敗した場合に発生する例外クラスです。

8.2.6 サンプルプログラム

```
# coding: utf-8

require "field_reports"

$param = {
  "template": {"paper": "A4"},
  "context": {
    "hello": {
      "new": "Tx",
      "value": "Hello, World!",
      "rect": [100, 700, 400, 750]
    }
  }
}

if ARGV.length == 1 then
  reports = FieldReports::Bridge.create_proxy()
  pdf = reports.render($param)
  File.binwrite(ARGV[0], pdf)
else
  p "usage: %s <outfile>" % $0
end
```

コード 8.2: hello.rb

8.3 PHP Bridge API

モジュール名 FieldReports

8.3.1 FieldReports\Bridge クラス

Field Reports と連携するための Proxy オブジェクトを生成します。

`create_proxy($uri = null)`

URI に応じた Field Reports Proxy オブジェクトを返却します。

引数

`$uri` Field Reports との接続方法を示す URI。

`null` を指定または省略した場合、環境変数 'REPORTS_PROXY' から URI を取得します。

環境変数 'REPORTS_PROXY' も未設定の場合の既定値は "exec:reports" です。

書式 (コマンド連携時)

```
exec:{exePath}?cwd={cwd}&loglevel={logLevel}
```

- cwd, loglevel は省略可能です。
- loglevel が 0 より大きい場合、標準エラー出力にログを出力します。

書式 (HTTP 連携時)

```
http://{hostName}:{portNumber}/
```

返り値

Proxy Field Reports Proxy オブジェクト

サンプルコード (コマンド連携時)

```
$reports = FieldReports\Bridge::create_proxy("exec:/usr/local/bin/reports?cwd=/usr/
share&loglevel=3");
```

サンプルコード (HTTP 連携時)

```
$reports = FieldReports\Bridge::create_proxy("http://localhost:50080/");
```

```
create_exec_proxy($exe_path="reports", $cwd=". ", $loglevel=0, $logout=null)
```

コマンド呼び出しにより Field Reports と連携する Proxy オブジェクトを生成します。

引数

\$exe_path (string) Field Reports コマンドのパス

\$cwd (string) Field Reports プロセス実行時のカレントディレクトリ

\$loglevel (int) ログ出力レベル (0: ログを出力しない, 1: ERROR ログ, 2: WARN ログ, 3: INFO ログ, 4: DEBUG ログ)

\$logout ログ出力先 Stream

返り値

Proxy Field Reports Proxy オブジェクト

```
create_http_proxy($base_address="http://localhost:50080/")
```

HTTP 通信により Field Reports と連携する Proxy オブジェクトを生成します。

引数

\$base_address (string) ベース URI

返り値

Proxy Field Reports Proxy オブジェクト

8.3.2 FieldReports\Proxy インターフェース

version()

バージョン番号を取得します。

返り値

string バージョン番号

例外

ReportsException Field Reports との連携に失敗した場合に発生

`render($param)`

レンダリング・パラメータを元にレンダリングを実行します。

引数

`$param (string|array)` JSON 文字列または連想配列形式レンダリング・パラメータ ([5 参照](#))

返り値

`string` PDF データ

例外

ReportsException Field Reports との連携に失敗した場合に発生

`parse($pdf)`

PDF データを解析し、フィールドや注釈の情報を取得します。

引数

`pdf (string)` PDF データ

返り値

`array` 解析結果

例外

ReportsException Field Reports との連携に失敗した場合に発生

8.3.3 ExeProxy クラス

コマンド呼び出しにより Field Reports との連携を行う、Proxy インターフェースの実装クラスです。

8.3.4 HttpProxy クラス

HTTP 通信により Field Reports との連携を行う、Proxy インターフェースの実装クラスです。

8.3.5 ReportsException クラス

Field Reports との連携に失敗した場合に発生する例外クラスです。

8.3.6 サンプルプログラム

```
<?php
require('vendor/autoload.php');

$param = [
    "template" => ["paper" => "A4"],
    "context" => [
        "hello" => [
            "new" => "Tx",
            "value" => "Hello, World!",
            "rect" => [100, 700, 400, 750]
        ]
    ]
];

$reports = FieldReports\Bridge::create_proxy();
$fp = fopen($argv[1], 'w');
fwrite($fp, $reports->render($param));
?>
```

コード 8.3: hello.php

8.4 Java VM Bridge API

パッケージ名 jp.co.field_works.field_reports

8.4.1 jp.co.field_works.field_reports.Bridge クラス

Field Reports と連携するための Proxy オブジェクトを生成します。

```
static Proxy createProxy(String uri)
URI に応じた Field Reports Proxy オブジェクトを返却します。
引数
uri Field Reports との接続方法を示す URI。
null を指定した場合、環境変数 'REPORTS_PROXY' から URI を取得します。
環境変数 'REPORTS_PROXY' も未設定の場合の既定値は"exec:reports"です。
書式 (コマンド連携時)
exec:{exePath}?cwd={ cwd }&loglevel={ logLevel }


- cwd, loglevel は省略可能です。
- loglevel が 0 より大きい場合、標準エラー出力にログを出力します。


書式 (HTTP 連携時)
```

`http://{hostName}:{portNumber}/`

返り値

Proxy Field Reports Proxy オブジェクト

サンプルコード（コマンド連携時）

```
import jp.co.field_works.field_reports.*;
Proxy reports = Bridge.createProxy("exec:/usr/local/bin/reports?cwd=/usr/share&
loglevel=3");
```

サンプルコード（HTTP 連携時）

```
import jp.co.field_works.field_reports.*;
Proxy reports = Bridge.createProxy("http://localhost:50080/");
```

`static Proxy createExecProxy(String exePath, String cwd, int logLevel, OutputStream
logStream)`

コマンド呼び出しにより Field Reports と連携する Proxy オブジェクトを生成します。

引数

exePath Field Reports コマンドのパス
cwd Field Reports プロセス実行時のカレントディレクトリ
loglevel ログ出力レベル（0: ログを出力しない, 1: ERROR ログ, 2: WARN ログ, 3: INFO ログ, 4: DEBUG ログ）
logStream ログ出力先 Stream

返り値

Proxy Field Reports Proxy オブジェクト

`static Proxy createHttpProxy(String baseUrl)`

HTTP 通信により Field Reports と連携する Proxy オブジェクトを生成します。

引数

baseUrl ベース URI

返り値

Proxy Field Reports Proxy オブジェクト

8.4.2 jp.co.field_works.field_reports.Proxy インターフェース

`String version()`

バージョン番号を取得します。

返り値

String バージョン番号

例外

ReportsException Field Reports との連携に失敗した場合に発生

```
byte[] render(String param)
レンダリング・パラメータを元にレンダリングを実行します。
引数
param (String) JSON 文字列形式レンダリング・パラメータ (5 参照)
返り値
byte[ ] PDF データ
例外
ReportsException Field Reports との連携に失敗した場合に発生

String parse(byte[] pdf)
PDF データを解析し、フィールドや注釈の情報を取得します。
引数
pdf (byte[ ]) PDF データ
返り値
String 解析結果 (JSON 形式)
例外
ReportsException Field Reports との連携に失敗した場合に発生
```

8.4.3 jp.co.field_works.field_reports.ExeProxy クラス

コマンド呼び出しにより Field Reports との連携を行う、Proxy インターフェースの実装クラスです。

8.4.4 jp.co.field_works.field_reports.HttpProxy クラス

HTTP 通信により Field Reports との連携を行う、Proxy インターフェースの実装クラスです。

8.4.5 ReportsException クラス

Field Reports との連携に失敗した場合に発生する例外クラスです。

8.4.6 サンプルプログラム

```
package example;

import java.io.FileOutputStream;
import jp.co.field_works.field_reports.*;

public class Hello
{
    public static void main(String[] args)
    {
        String param = "{\n"
        "  \"template\": {\"paper\": \"A4\"},\n"
        "  \"context\": {\n"
        "    \"hello\": {\n"
        "      \"new\": \"Tx\", \n"
        "      \"value\": \"Hello, World!\", \n"
        "      \"rect\": [100, 700, 400, 750]\n"
        "    }\n"
        "  }\n"
        "}\n";
        Proxy reports = Bridge.createProxy(null);
        try {
            FileOutputStream fos = new FileOutputStream(args[0]);
            byte[] pdf = reports.render(param);
            fos.write(pdf);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

コード 8.4: Hello.java

8.5 .NET Bridge API

アセンブリ名 FieldWorks.FieldReports

8.5.1 FieldWorks.FieldReports.Bridge クラス

Field Reports と連携するための Proxy オブジェクトを生成します。

```
static IProxy CreateProxy(string uriString=null)
URI に応じた Field Reports Proxy オブジェクトを返却します。
引数
```

`uriString` Field Reports との接続方法を示す URI。

`null` を指定または省略した場合、環境変数 ‘REPORTS_PROXY’ から URI を取得します。

環境変数 ‘REPORTS_PROXY’ も未設定の場合の既定値は “exec:reports” です。

書式 (コマンド連携時)

```
exec:{exePath}?cwd={ cwd}&loglevel={logLevel}
```

- `cwd`, `loglevel` は省略可能です。
- `loglevel` が 0 より大きい場合、標準エラー出力にログを出力します。

書式 (HTTP 連携時)

```
http://{hostName}:{portNumber}/
```

返り値

Field Reports Proxy オブジェクト

サンプルコード (コマンド連携時)

```
using FieldWorks.FieldReports;  
var reports = Bridge.createProxy("exec:/usr/local/bin/reports?cwd=/usr/share&loglev  
el=3");
```

サンプルコード (HTTP 連携時)

```
using FieldWorks.FieldReports;  
var reports = Bridge.createProxy("http://localhost:50080/");  
  
static IProxy CreateExecProxy(string exePath="reports", string cwd=". ", int logLevel=0,  
TextWriter logWriter=null)
```

コマンド呼び出しにより Field Reports と連携する Proxy オブジェクトを生成します。

引数

`exePath` Field Reports コマンドのパス
`cwd` Field Reports プロセス実行時のカレントディレクトリ
`logLevel` ログ出力レベル (0: ログを出力しない, 1: ERROR ログ, 2: WARN ログ, 3: INFO ロ
グ, 4: DEBUG ログ)
`logWriter` ログ出力先 Stream

返り値

Field Reports Proxy オブジェクト

```
static IProxy CreateHttpProxy(string baseUri="http://localhost:50080/")  
HTTP 通信により Field Reports と連携する Proxy オブジェクトを生成します。
```

引数

`baseUri` ベース URI

返り値

Field Reports Proxy オブジェクト

```
static IProxy CreateHttpProxy(HttpClient httpClient)  
HTTP 通信により Field Reports と連携する Proxy オブジェクトを生成します。
```

引数で指定した HttpClient を利用して、HTTP 通信を行います。httpClient には、あらかじめ BaseAddress を設定してください。

引数

httpClient HttpClient インスタンス

返り値

Field Reports Proxy オブジェクト

8.5.2 FieldWorks.FieldReports.IProxy インターフェース

string Version()

バージョン番号を取得します。

返り値

バージョン番号

例外

ReportsException Field Reports との連携に失敗した場合に発生

byte[] Render(object param)

レンダリング・パラメータを元にレンダリングを実行します。

引数

param JSON 文字列またはシリализ可能なオブジェクト形式レンダリング・パラメータ (5
参照)

返り値

PDF データ

例外

ReportsException Field Reports との連携に失敗した場合に発生

Task<byte[]> RenderAsync(object param)

レンダリング・パラメータを元にレンダリングを非同期で実行します。

引数

param JSON 文字列またはシリализ可能なオブジェクト形式レンダリング・パラメータ (5
参照)

返り値

PDF データ

例外

ReportsException Field Reports との連携に失敗した場合に発生

string Parse(byte[] pdf)

PDF データを解析し、フィールドや注釈の情報を取得します。

引数

pdf PDF データ

返り値

解析結果 (JSON 文字列形式)

例外

ReportsException Field Reports との連携に失敗した場合に発生
Task<string> ParseAsync(byte[] pdf)

PDF データを非同期で解析し、フィールドや注釈の情報を取得します。

引数

pdf PDF データ

返り値

解析結果 (JSON 文字列形式)

例外

ReportsException Field Reports との連携に失敗した場合に発生

8.5.3 FieldWorks.FieldReports.ExeProxy クラス

コマンド呼び出しにより Field Reports との連携を行う、 IProxy インターフェースの実装クラスです。

8.5.4 FieldWorks.FieldReports.HttpProxy クラス

HTTP 通信により Field Reports との連携を行う、 IProxy インターフェースの実装クラスです。

8.5.5 ReportsException クラス

Field Reports との連携に失敗した場合に発生する例外クラスです。

```

using System;
using System.IO;
using FieldWorks.FieldReports;

static class Example
{
    static void Main(string[] args)
    {
        var param = new {
            template = new {
                paper = "A4"
            },
            context = new {
                hello = new {
                    @new = "Tx",
                    value = "Hello, World!",
                    rect = new int[] {100, 700, 400, 750}
                }
            }
        };
        var reports = Bridge.CreateProxy();
        var pdf = reports.Render(param);
        using (var fs = new FileStream(args[0], FileMode.Create))
        using (var bw = new BinaryWriter(fs))
        {
            bw.Write(pdf);
        }
    }
}

```

コード 8.5: Program.cs

8.6 コマンドライン I/F

Field Reports のコマンドライン・プログラムは、以下の書式で呼び出します。

reports <サブコマンド> [<オプション>] [<引数>]

以下のサブコマンドが利用できます。

version

Field Reports のバージョンを表示します。

render <入力ファイル名> <出力ファイル名>

PDF 帳票を生成します。<入力ファイル名>には、JSON 形式で記述したレンダリングパラメータのファイル名を指定します。“-”を指定した場合、標準入力からレンダリングパラメータを取得します。<出力ファイル名>には、PDF 帳票を保存するファイル名を指定します。“-”を指定した場合、生成した PDF の内容を標準出力へ出力します。

parse <入力ファイル名>

PDF を解析して、PDF の構造を JSON 形式で出力します。<入力ファイル名> には、解析対象となる PDF のファイル名を指定します。“-” を指定した場合、標準入力からレンダリングパラメータを取得します。

font <フォントファイル名> ...

指定したフォントファイルの情報を表示します。<フォントファイル名> は、スペースで区切って複数指定することができます。--json オプションを指定すると、リソース定義要素の雛形を JSON 形式で出力します。

check

登録されたライセンスキーが有効かどうか確認します。

server

Field Reports をサーバーモードで起動します。サーバーモードでは、[8.7](#) に示す Web API を待ち受けます。ポート番号は、--port オプションで変更できます（既定値：50080）。

各サブコマンドで有効なオプションについては、以下のコマンドを実行して確認してください。

```
% reports <サブコマンド> --help
```

8.7 Web API

サーバーモードで利用可能な Web API の一覧を表 [8.2](#) に示します。

表 8.2: Web API 一覧

URL	メソッド	送信データ	レスポンス	説明
/version	GET	-	バージョン番号	Field Reports のバージョン番号を返却します。
/render	POST	レンダリング パラメータ	PDF	JSON 形式のレンダリングパラメータを元にレンダリングを実行し、結果をバイト文字列で返却します。
/parse	POST	PDF	JSON	PDF データを解析し、フィールドや注釈の情報を取得します。解析結果は、JSON 形式の文字列で返されます。

8.7.1 /version

Field Reports のバージョン番号を文字列で返却します。

リクエスト

■HTTP リクエスト

```
GET http://<ホスト名>:<ポート番号>/version
```

応答

■メッセージボディー

バージョン番号を文字列として返却します。

8.7.2 /render

JSON 形式のレンダリングパラメータを元にレンダリングを実施し、生成した PDF をバイト文字列で返却します。

リクエスト

■HTTP リクエスト

POST http://<ホスト名>:<ポート番号>/render

■HTTP ヘッダー

HTTP ヘッダー	値	説明
Content-Type	application/json	コンテンツ種別

■メッセージボディー

レンダリングパラメータ ([5 参照](#))

■リクエスト例

```
POST http://localhost:50080/render
Content-Type: application/json

{
    "template": {"paper": "A4"},
    "context": {
        "hello_1": {
            "new": "Tx",
            "value": "Hello, World!",
            "rect": [100, 700, 400, 750],
            "font": "/Times-Roman"
        },
        "hello_2": {
            "new": "Tx",
            "value": "Hello, World!",
            "rect": [100, 600, 400, 650],
            "font": "/Helvetica-Oblique"
        }
    }
}
```

応答

■HTTP 応答ヘッダー

HTTP ヘッダー	値	説明
Content-Type	application/pdf	コンテンツ種別

■メッセージボディー

PDF をバイト文字列として返却します。

8.7.3 /parse

PDF データを解析し、フィールドや注釈の情報を取得します。解析結果は、JSON 形式の文字列で返されます。

リクエスト

■HTTP リクエスト

POST http://<ホスト名>:<ポート番号>/parse

■HTTP ヘッダー

HTTP ヘッダー	値	説明
Content-Type	application/pdf	コンテンツ種別

■メッセージボディー

PDF をバイト文字列として送信します。

応答

■HTTP 応答ヘッダー

HTTP ヘッダー	値	説明
Content-Type	application/json	コンテンツ種別

■メッセージボディー

JSON 文字列として返却します。

■応答例

```
HTTP/1.1 200 OK
content-length: 28753
content-type: application/json
```

```
{  
  "template": {  
    "media-box": [ [ 0.0, 0.0, 595.0, 842.0 ] ],  
    "rotate": [ 0 ]  
  },  
  "context": {  
    "date": {  
      "type": "Tx",  
      "rect": [ 459.2, 769.425, 555.016, 786.439 ],  
      "page": 0,  
      "font": "/KozMinProVI-Regular",  
      "font-size": 10.0,  
      "color": [ 0.0 ],  
      "text-stroke-color": [],  
      "value": "",  
      "multiline": false,  
      "text-align": "Right",  
      "border-color": [],  
      "background-color": [],  
      "rotate": 0  
    },  
    ...  
  },  
  "annotation": []  
}
```

付録 A

利用ライブラリ

本ソフトウェアで利用しているライブラリの一覧を表 A.1 に示します。

表 A.1: 利用しているライブラリの一覧

名称	版数	開発元情報・ライセンス条件等
OCaml 標準ライブラリ	4.13.1	Institut National de Recherche en Informatique et en Automatique (INRIA) http://caml.inria.fr/ocaml/ LGPL with linking exceptions
CamlPDF	2.3	Coherent Graphics Ltd. http://www.coherentpdf.com/ BSD licence with special exceptions
ExtLib	1.7.8	Nicolas Cannasse 他 https://github.com/ygrek/ocaml-extlib LGPL-2.1-only with OCaml-LGPL-linking-exception
json-wheel	1.0.6	Wink Technologies, Inc., Martin Jambon https://mjambon.github.io/mjambon2016/json-wheel.html BSD license
cryptokit	1.14	Xavier Leroy. http://forge.ocamlcore.org/projects/cryptokit/ GNU Lesser General Public with static compilation exception
cohttp	5.0.0	Anil Madhavapeddy 他 https://github.com/mirage/ocaml-cohttp ISC
GNU C Library	2.17	Free Software Foundation, Inc. https://www.gnu.org/software/libc GNU LGPL v2.1
GMP	6.0.0	Free Software Foundation, Inc. http://gmplib.org/ GNU LGPL v3
libev	4.15	Marc Lehmann and Emanuele Giaquinta http://software.schmorp.de/pkg/libev BSD

付録 B

文字参照

B.1 Lantin 1 Characters

表 B.1: 文字実体参照 (Lantin 1 Characters)

文字	文字実体参照	説明
	 	改行禁止スペース
¡	¡	反転させた感嘆符
¢	¢	セント
£	£	ポンド
¤	¤	通貨
¥	¥	円
׀	¦	縦破線
§	§	セクション
„	¨	ウムラウト
©	©	著作権
ª	ª	女性序数
«	«	二重山括弧 (開始)
»	¬	ノット、角ダッシュ
‐	­	ソフトハイフン
®	®	登録商標
‐	¯	長音記号
°	°	度
±	±	プラスマイナス
²	²	2乗 (2の上付き文字)
³	³	3乗 (3の上付き文字)
ˊ	´	鋭 (揚音) アクセント
µ	µ	マイクロ
¶	¶	パラグラフ
·	·	中黒
,	¸	セディーユ
¹	¹	1乗 (1の上付き文字)
º	º	男性序数
»	»	二重山括弧 (終了)
¼	¼	4分の1
½	½	2分の1
¾	¾	4分の3

表 B.1: 文字実体参照 (Lantin 1 Characters : 続き)

文字	文字実体参照	説明
ÿ	¿	反転させた疑問符
À	À	低（抑音）アクセントつき A
Á	Á	鋭（揚音）アクセントつき A
Â	Â	曲折アクセントつき A
Ã	Ã	チルダつき A
Ä	Ä	ウムラウトつき A
Å	Å	リングつき A
Æ	Æ	連字の AE
Ç	Ç	セディーウつき C
È	È	低（抑音）アクセントつき E
É	É	鋭（揚音）アクセントつき E
Ê	Ê	曲折アクセントつき E
Ë	Ë	ウムラウトつき E
Ì	Ì	低（抑音）アクセントつき I
Í	Í	鋭（揚音）アクセントつき I
Î	Î	曲折アクセントつき I
Ï	Ï	ウムラウトつき I
Ð	Ð	古英語のエズ (ETH)
Ñ	Ñ	チルダつき N
Ò	Ò	低（抑音）アクセントつき O
Ó	Ó	鋭（揚音）アクセントつき O
Ô	Ô	曲折アクセントつき O
Õ	Õ	チルダつき O
Ö	Ö	ウムラウトつき O
×	×	乗法、かけ算
Ø	Ø	スラッシュつき O
Ù	Ù	低（抑音）アクセントつき U
Ú	Ú	鋭（揚音）アクセントつき U
Û	Û	曲折アクセントつき U
Ü	Ü	ウムラウトつき U
Ý	Ý	鋭（揚音）アクセントつき Y
Þ	Þ	古英語のソーン (THORN)
ß	ß	連字の sz (ドイツ語など)
à	à	低（抑音）アクセントつき a
á	á	鋭（揚音）アクセントつき a
â	â	曲折アクセントつき a
ã	ã	チルダつき a
ä	ä	ウムラウトつき a
å	å	リングつき a
æ	æ	連字の ae
ç	ç	セディーウつき c
è	è	低（抑音）アクセントつき e
é	é	鋭（揚音）アクセントつき e
ê	ê	曲折アクセントつき e
ë	ë	ウムラウトつき e

表 B.1: 文字実体参照 (Lantin 1 Characters : 続き)

文字	文字実体参照	説明
ѝ	ì	低（抑音）アクセントつき i
ѝ	í	鋭（揚音）アクセントつき i
ѝ	î	曲折アクセントつき i
ѝ	ï	ウムラウトつき i
ð	ð	古英語のエズ (eth)
ñ	ñ	チルダつき n
ò	ò	低（抑音）アクセントつき o
ó	ó	鋭（揚音）アクセントつき o
ô	ô	曲折アクセントつき o
õ	õ	チルダつき o
ö	ö	ウムラウトつき o
÷	÷	除法、割り算
ø	ø	チルダつき o
ù	ù	低（抑音）アクセントつき u
ú	ú	鋭（揚音）アクセントつき u
û	û	曲折アクセントつき u
ü	ü	ウムラウトつき u
ý	ý	鋭（揚音）アクセントつき y
þ	þ	古英語のソーン (thorn)
ÿ	ÿ	ウムラウトつき y

B.2 Special Characters

表 B.2: 文字実体参照 (Special Characters)

文字	文字実体参照	説明
"	"	二重引用符
&	&	アンド (アンパサンド)
<	<	小なり
>	>	大なり
'	'	アポストロフィ
Œ	Œ	連字の OE
œ	œ	連字の oe
Š	Š	キャロンつき S
š	š	キャロンつき s
Ÿ	Ÿ	ウムラウトつき Y
(U+02C6)	ˆ	曲折アクセント
~	˜	チルダ
	 	n 文字幅スペース
	 	m 文字幅スペース
	 	細いスペース
	‌	ゼロ幅ノンジョイナー
	‍	ゼロ幅ジョイナー
	‎	左から右マーク
	‏	右から左マーク
-	–	n 文字幅ダッシュ
—	—	m 文字幅ダッシュ
'	‘	引用符 (開始)
,	’	引用符 (終了)
,	‚	下付き引用符
"	“	二重引用符 (開始)
"	”	二重引用符 (終了)
"	„	下付き二重引用符
†	†	参照符
‡	‡	二重参照符
%	‰	千分率 (パーミレージ)
<	‹	山括弧 (開始)
>	›	山括弧 (終了)
€	€	ユーロ

B.3 Symbols

表 B.3: 文字実体参照 (Symbols)

文字	文字実体参照	説明
A	Α	アルファ
B	Β	ベータ
Г	Γ	ガンマ
Δ	Δ	デルタ
Ε	Ε	イプシロン
Ζ	Ζ	ゼータ
Η	Η	イータ
Θ	Θ	シータ
Ι	Ι	イオタ
Κ	Κ	カッパ
Λ	Λ	ラムダ
Μ	Μ	ミュー
Ν	Ν	ニュー
Ξ	Ξ	クシー (クサイ)
Ο	Ο	オミクロン
Π	Π	パイ
Ρ	Ρ	ロー
Σ	Σ	シグマ
Τ	Τ	タウ
Υ	Υ	ユプシロン
Φ	Φ	ファイ
Χ	Χ	キー (カイ)
Ψ	Ψ	プシー (プサイ)
Ω	Ω	オメガ
α	α	アルファ
β	β	ベータ
γ	γ	ガンマ
δ	δ	デルタ
ε	ε	イプシロン
ζ	ζ	ゼータ
η	η	イータ
θ	θ	シータ
ι	ι	イオタ
κ	κ	カッパ
λ	λ	ラムダ
μ	μ	ミュー
ν	ν	ニュー
ξ	ξ	クシー (クサイ)
ο	ο	オミクロン
π	π	パイ
ρ	ρ	ロー
ς	ς	ファイナルシグマ
σ	σ	シグマ
τ	τ	タウ
υ	υ	ユプシロン

表 B.3: 文字実体参照 (Symbols : 続き)

文字	文字実体参照	説明
Φ	φ	ファイ
χ	χ	キー (カイ)
Ψ	ψ	プシー (プサイ)
ω	ω	オメガ
(U+03D1)	ϑ	シータシンボル
(U+03D2)	ϒ	フックつきユプシロン
(U+03D6)	ϖ	パイシンボル
•	•	ブリット (中黒)
…	…	三点リーダー
'	′	プライム符号 (分またはフィート)
"	″	二重プライム符号 (秒またはインチ)
(U+203E)	‾	オーバーライン (オーバースコア)
(U+2044)	⁄	分数のスラッシュ
(U+2118)	℘	手書き風の P
(U+2111)	ℑ	手書き風の I (虚数の I)
(U+211C)	ℜ	手書き風の R (実数の R)
™	™	商標
ℵ	ℵ	アーレフ (第一超限基数)
←	←	左矢印
↑	↑	上矢印
→	→	右矢印
↓	↓	下矢印
↔	↔	左右矢印
(U+21B5)	↵	改行キー (キャリッジリターン)
⇐	⇐	二重左矢印
(U+21D1)	⇑	二重上矢印
⇒	⇒	二重右矢印
(U+21D3)	⇓	二重下矢印
⇒	⇔	二重左右矢印
∀	∀	すべての (数学記号)
∂	∂	偏微分 (数学記号)
∃	∃	存在する (数学記号)
∅	∅	空集合 (数学記号)
∇	∇	ナブラ (数学記号)
∈	∈	要素として含まれる (数学記号)
∉	∉	要素として含まれない (数学記号)
⊑	∋	元として含む (数学記号)
(U+220F)	∏	n の積 (数学記号)
Σ	∑	n の総和 (数学記号)
-	−	マイナス (数学記号)
*	∗	アスタリスク (数学記号)
√	√	平方根 (数学記号)
∞	∝	比例 (数学記号)
∞	∞	無限 (数学記号)
∠	∠	角度 (数学記号)

表 B.3: 文字実体参照 (Symbols : 続き)

文字	文字実体参照	説明
∧	∧	かつ (数学記号)
∨	∨	または (数学記号)
∩	∩	積集合 (数学記号)
∪	∪	和集合 (数学記号)
∫	∫	積分 (数学記号)
∴	∴	ゆえに (数学記号)
~	∼	チルダ (数学記号)
≈	≅	およそ等しい (数学記号)
≈	≈	ほぼ等しい (数学記号)
≠	≠	等しくない (数学記号)
≡	≡	合同 (数学記号)
(U+2264)	&le	小なりイコール (数学記号)
(U+2265)	≥	大なりイコール (数学記号)
⊂	⊂	部分集合 (含まれる) (数学記号)
⊃	⊃	部分集合 (含む) (数学記号)
⊄	⊄	部分集合 (含まれない) (数学記号)
⊆	⊆	部分集合 (含まれるか等しい) (数学記号)
⊇	⊇	部分集合 (含むか等しい) (数学記号)
⊕	⊕	丸つき加算記号
⊗	⊗	丸つき乗算記号 (ベクトル積)
⊥	⊥	垂直 (数学記号)
.	⋅	ドット
(U+2308)	⌈	左シーリング
(U+2309)	⌉	右シーリング
(U+230A)	⌊	左フロアー
(U+230B)	⌋	右フロアー
(U+2329)	⟨	左アングル
(U+232A)	⟩	右アングル
(U+25CA)	◊	ひし形
♠	♠	スペード
♣	♣	クラブ
♥	♥	ハート
♦	♦	ダイアモンド

付録 C

1.5 版からの変更点

C.1 フィールド辞書

C.1.1 text-align と vertical-align

縦書き時の text-align, vertical-align の意味を変更しました。

書式モード（横組み／縦組み）にかかわらず、text-align は 1 行内のテキストの整列方向を指定する属性、vertical-align は行の整列方法を指定する属性とします。

multiline 属性が有効であっても vertical-align の指定は有効です。

表 C.1: field 辞書での変更点

キー	型	値
text-align	列挙値	(オプション) テキストの整列方法を指定します。 Left 左寄せ（縦組みの場合上寄せ） Center 中央寄せ Right 右寄せ（縦組みの場合下寄せ） Justify 均等割付 初期値 : Left.
vertical-align	列挙値	(オプション) 行の整列方法を指定します。 Top 上寄せ（縦組みの場合右寄せ） Middle 中央寄せ Bottom 下寄せ（縦組みの場合左寄せ） 初期値 : Top.

C.1.2 padding

2.0 からは、マイナスのパディング値を指定することができます。

C.1.3 normalize

廃止しました。

C.2 API

C.2.1 言語 Bridge

共有ライブラリを介したインプロセスでの連携方式を廃止し、コマンド呼び出しましたは HTTP 通信による連携方式としました。

C.2.2 コマンドライン I/F

サブコマンド名を変更しました。

create → render

C.2.3 低レベル I/F

廃止しました。