PDF 出力ライブラリ仕様書

Version 3.0 for Windows

第 1.4 版

2014/8/19

アンテナハウス株式会社

改訂履歴

2011 年 6 月 28 日

初版

2011 年 11 月 11 日

第 1.1 版

・8. 7. 31. フレーム番号設定関数
関数名の誤りを修正

・8. 7. 32. フレーム番号設定関数(ハンドル指定)
関数名の誤りを修正

・10. イメージ出力について

TIFF 形式の制限事項を修正

・著作権情報を改訂

2012 年 4 月 20 日

第 1.2 版

・１．はじめに
動作環境から Windows2000、Linux、Unix の記述を削除

・1. 1. コンパイラ
再頒布モジュールの記述を追加

2014 年 7 月 29 日

第 1.3 版

・１．はじめに
動作環境へ 64bitOS の記述を追加

・8. 5. 9. ペイント関数
透明度オプション（transparency）の説明を追加

2014 年 8 月 19 日

第 1.4 版

・１．はじめに
動作環境から Windows XP の記述を削除

・12. 2. 著作権情報
ICU の情報を改訂

1. はじめに 1
1.1. コンパイラ 1
2. 機能概要 1
2.1. テキスト 1
2.2. イメージ 2
2.3. グラフィックス 2
2.4. シェーディング 2
2.5. 関数 2
2.6. アウトライン 2
2.7. 注釈 2
2.8. セキュリティ 2
2.9. PDF/X 3
2.10. PDF/A 3
3. 構成ファイル 3
3.1. フォント設定ファイル 3
3.2. ライブラリファイル 3
4. 戻り値 4
5. 制限事項 8
5.1. PDF バージョン 8
5.2. フォント 8
5.3. カラースペース 8
5.4. イメージ 8
5.5. その他 8
5.6. PDF/X に関する注意事項 8
5.7. PDF/A に関する注意事項 9
6. 共通データ形式 10
6.1. PL_PointTD 10
6.2. PL_RectangleTD 10
6.3. PL_LinePatternTD 10
6.4. PL_LineTD 11
6.5. PL_BorderTD 11
6.6. PL_ColorSpaceTD 12
6.7. PL_ColorTD 14
6.8. PL_DestTD 15
6.9. UCHAR_t 16
7. 関数一覧 17
8. 関数説明 20
8.1. 基本関数 20
8.1.1. 初期化関数 20
8.1.2. 終了関数 20
8.1.3. 強制終了関数 20
8.1.4. PDF ファイルオープン関数 20
8.1.5. PDF ファイルオープン関数(ストリーム) 21
8.1.6. PDF クローズ関数 21
8.1.7. フォントデータ初期化関数 22
8.1.8. フォントデータ終了関数 22
8.1.9. PDF ファイル一時クローズ関数 22

8.1.10. PDF ファイル再オープン関数 23
8.2. オプション設定関数 23
8.2.1. PDF バージョン設定関数 23
8.2.2. セキュリティ設定関数 24
8.2.3. 圧縮設定関数 27
8.2.4. ASCII 形式出力設定関数 28
8.2.5. 画像比較設定関数 29
8.2.6. オブジェクト圧縮設定関数 29
8.2.7. 文書情報設定関数 29
8.2.8. ベース URI 設定関数 30
8.2.9. ページモード設定関数 30
8.2.10. ページレイアウト設定関数 31
8.2.11. ビューアプレファレンス設定関数 32
8.2.12. オープンアクション設定関数 34
8.2.13. オープンアクション設定関数 2 35
8.2.14. 事前定義された名前によるオープンアクション設定関数 35
8.2.15. ページラベル設定関数 35
8.2.16. 名前付き宛先定義関数 36
8.2.17. PDF/X エラー処理設定設定関数 36
8.2.18. 言語設定関数 37
8.2.19. タグ付 PDF エラー処理設定関数 37
8.2.20. タグ付 PDF 設定関数 38
8.2.21. 埋め込みフォント設定関数 38
8.2.22. ミッシンググリフ処理設定関数 40
8.2.23. Identity 設定関数 41
8.2.24. イメージダウンサンプリング設定関数 41
8.2.25. 透過イメージの処理方法の設定関数 42
8.2.26. イメージ（PNG、GIF、TIFF、JPG）のパススルー抑止設定関数 43
8.2.27. αチャネル付き 1 ビット GIF イメージの変換設定関数 44
8.2.28. カラープロファイル付きイメージの処理モード設定関数 44
8.2.29. ヘッダコメント設定関数 45
8.2.30. ヘッダコメント設定関数（ストリーム） 45
8.2.31. XMP データ設定関数 46
8.2.32. XMP データ設定関数（ストリーム） 46
8.2.33. PDF/A エラー処理設定設定関数 46
8.2.34. PDF/A 出力時のカラースペース自動変換関数 47
8.2.35. PDF/X 出力時のカラースペース自動変換関数 47
8.3. ページ制御関数 47
8.3.1. ページツリーノード作成関数 47
8.3.2. ページツリーノードクローズ関数 48
8.3.3. ページオブジェクト作成関数 48
8.3.4. ページオブジェクトクローズ関数 48
8.3.5. カレントページ用紙サイズ設定関数 49
8.3.6. クロップボックスオフセット設定関数 49
8.3.7. ブリードボックスオフセット設定関数 50
8.3.8. トリムボックスオフセット設定関数 50
8.3.9. アートボックスオフセット設定関数 50
8.3.10. 境界ボックス設定関数 51
8.3.11. ページ逆順出力設定関数 51
8.3.12. ページ逆順出力設定関数（ページ範囲指定） 52
8.4. グラフィックス状態設定関数 52
8.4.1. グラフィックス状態退避関数 52
8.4.2. グラフィックス状態復旧関数 52
8.4.3. 変換行列設定関数 52
8.4.4. 線幅設定関数 53
8.4.5. ラインキャップスタイル設定関数 53
8.4.6. ラインジョインスタイル設定関数 54

8.4.7. マイターリミット設定関数 55
8.4.8. 線パターン設定関数 55
8.4.9. レンダリングインテント設定関数 56
8.4.10. グラフィックス状態設定関数 56
8.4.11. カラー設定関数 58
8.4.12. カラースペースロード関数 59
8.4.13. カラースペース設定関数 60
8.4.14. カラープロファイルロード関数 60
8.4.15. カラープロファイルロード関数(ストリーム) 61
8.4.16. カラープロファイルロード関数(メモリバッファエリア) 61
8.4.17. デフォルトカラースペース設定関数 61
8.4.18. カラープロファイル検査関数 62
8.4.19. カラープロファイル検査関数(ストリーム) 63
8.4.20. カラープロファイル検査関数(メモリバッファエリア) 63
8.4.21. カラープロファイル情報取得関数 63
8.5. パス関連オペレータ出力関数 66
8.5.1. カレントポイント設定関数 66
8.5.2. 直線パス出力関数 66
8.5.3. 3 次ベジェ曲線パス出力関数 1 67
8.5.4. 3 次ベジェ曲線パス出力関数 2 67
8.5.5. 3 次ベジェ曲線パス出力関数 3 67
8.5.6. 円パス出力関数 68
8.5.7. パスクローズ関数 68
8.5.8. 矩形パス出力関数 69
8.5.9. ペイント関数 69
8.5.10. クリッピングパス設定関数 70
8.6. テキストオペレータ出力関数 70
8.6.1. フォント設定関数 70
8.6.2. 記述方向設定関数 71
8.6.3. 変換行列設定関数 72
8.6.4. テキスト状態パラメータ設定関数 72
8.6.5. 出力位置設定関数 73
8.6.6. 改行出力関数 74
8.6.7. 文字列出力関数 74
8.6.8. 文字列出力関数 2 74
8.6.9. グリフ番号指定による文字列出力関数 76
8.6.10. 文字幅計算関数 77
8.6.11. 文字列表示幅計算関数 77
8.6.12. 文字列表示幅計算関数 2 78
8.6.13. ミッシンググリフ文字コード取得関数 78
8.6.14. ミッシンググリフフォント名取得関数 79
8.7. イメージ操作関数 79
8.7.1. イメージテスト関数 81
8.7.2. イメージマスク検査関数 81
8.7.3. イメージ初期化関数 82
8.7.4. イメージマスク初期化関数 82
8.7.5. イメージオープン関数 83
8.7.6. イメージオープン関数(ストリーム) 83
8.7.7. マスク付きイメージのオープン関数 83
8.7.8. マスク付きイメージのオープン関数(ストリーム) 84
8.7.9. イメージ出力関数 84
8.7.10. イメージクローズ関数 84
8.7.11. イメージ終了関数 85
8.7.12. グレースケールイメージ着色関数 85
8.7.13. イメージサイズ取得関数 85
8.7.14. イメージサイズ取得関数(ストリーム) 86
8.7.15. イメージカラープロファイル取得関数 86

8.7.16. イメージカラープロファイル取得関数(ファイル作成) 86
8.7.17. イメージカラープロファイル取得関数(ストリーム作成) 87
8.7.18. イメージ検査関数(Stream 版) 87
8.7.19. イメージマスク検査関数(Stream 版) 87
8.7.20. イメージのロード関数(Stream 版) 88
8.7.21. マスクイメージのロード関数(Stream 版) 88
8.7.22. マスク付きイメージのロード関数(Stream 版) 88
8.7.23. イメージ出力関数(Image Handle 指定) 89
8.7.24. イメージの削除関数(Stream 版) 89
8.7.25. イメージカラープロファイル取得関数(Image Handle 指定) 89
8.7.26. イメージカラープロファイル取得関数(Stream 作成, Image Handle 指定) 90
8.7.27. イメージサイズ取得関数(Image Handle 指定) 90
8.7.28. グレースケールイメージ着色関数(イメージハンドル指定) 90
8.7.29. フレーム数取得関数 91
8.7.30. フレーム数取得関数(ハンドル指定) 91
8.7.31. フレーム番号設定関数 91
8.7.32. フレーム番号設定関数(ハンドル指定) 92
8.7.33. カラーキー設定関数 92
8.7.34. カラーキー設定(ハンドル指定) 93
8.8. 関数オブジェクト出力関数 93
8.8.1. 関数オブジェクト初期化関数 93
8.8.2. 関数オブジェクト作成関数 95
8.8.3. 関数オブジェクトフリー関数 95
8.9. パターン定義関数 96
8.9.1. タイプ１パターン作成開始関数 96
8.9.2. タイプ１パターン作成終了関数 97
8.9.3. シェーディングオブジェクト作成関数 97
8.9.4. タイプ２パターン作成関数 99
8.9.5. パターン設定関数 100
8.9.6. シェーディング設定関数 100
8.10. アウトライン出力関数 101
8.10.1. アウトライン階層作成関数 101
8.10.2. アウトライン階層作成関数 2 101
8.10.3. リモートアウトライン階層作成関数 102
8.10.4. アウトライン項目作成関数 103
8.10.5. アウトライン項目作成関数 2 103
8.10.6. リモートアウトライン項目作成関数 104
8.10.7. アウトライン階層クローズ関数 104
8.11. 注釈オブジェクト出力関数 104
8.11.1. 注釈初期化関数 106
8.11.2. 注釈共通データ設定関数 107
8.11.3. 注釈終了関数 109
8.11.4. フリーテキスト注釈設定関数 109
8.11.5. テキスト注釈設定関数 110
8.11.6. ライン注釈設定関数 110
8.11.7. 正方形注釈設定関数 111
8.11.8. 円注釈設定関数 112
8.11.9. 多角形注釈設定関数 113
8.11.10. 折線注釈設定関数 113
8.11.11. ハイライト注釈設定関数 114
8.11.12. 下線注釈設定関数 115
8.11.13. 波線注釈設定関数 115
8.11.14. ストライクアウト注釈設定関数 116
8.11.15. ラバースタンプ注釈設定関数 116
8.11.16. キャレット注釈設定関数 117
8.11.17. インク注釈設定関数 117
8.11.18. ファイル添付注釈設定関数 118

8.11.19. サウンド注釈設定関数 119
8.11.20. ポップアップ注釈設定関数 120
8.11.21. ムービー注釈設定関数 121
8.11.22. リンク注釈(GoTo)設定関数 123
8.11.23. リンク注釈(GoTo)設定関数 2 124
8.11.24. リンク注釈(Launch/URI)設定関数 125
8.11.25. リンク注釈(GoToR)設定関数 125
8.11.26. アクション指定のリンク注釈 126
8.11.27. アクション指定のリンク注釈 2 126
8.12. PDF インポート 127
8.12.1. PDF インポート初期化関数 127
8.12.2. PDF インポート初期化関数(Stream 版) 128
8.12.3. インポート PDF ファイルテスト関数 128
8.12.4. インポート PDF ファイルの頁数取得関数 128
8.12.5. インポート PDF ファイルのバージョン取得関数 129
8.12.6. インポート PDF ファイルのセキュリティ情報取得関数 129
8.12.7. インポート PDF のページ設定関数 129
8.12.8. インポート PDF の用紙サイズ取得関数 130
8.12.9. インポート PDF の回転角度取得関数 130
8.12.10. PDF インポート実行関数 130
8.12.11. PDF インポート終了関数 131
8.12.12. インポート PDF のページ境界取得関数 131
8.12.13. インポート PDF のページ境界指定関数 132
8.12.14. 領域指定付き PDF インポート実行関数 132
8.12.15. インポート PDF のタグ情報検査関数 133
8.12.16. インポート PDF の OutputIntent の数の取得関数 133
8.12.17. インポート PDF の OutputIntent 取得関数 133
8.12.18. インポート PDF のファイル ID 取得関数 134
8.12.19. インポート PDF の有効注釈設定関数 134
8.12.20. インポート PDF の有効アクロフォーム設定関数 135
8.13. FormXObject 135
8.13.1. FormXObject 定義開始関数 136
8.13.2. FormXObject 定義終了関数 136
8.13.3. FormXObject 出力関数 136
8.13.4. ピースインフォ指定関数 136
8.14. Output Intent 設定関数 137
8.14.1. 標準出力インテントの Load 関数 137
8.14.2. 汎用出力インテントの Load 関数 139
8.14.3. Output Intent 設定関数 140
8.15. タグ付 PDF 140
8.15.1. Attribute の Load 機能 140
8.15.2. Marked Content の開始設定機能 145
8.15.3. Marked Content の終了指定機能 147
8.15.4. Marked Content リーフの開始設定機能 147
8.15.5. Marked Content 子節点の終了設定機能 147
8.15.6. Marked Content のアクティブにする設定機能 148
8.16. アクション作成関数 148
8.16.1. pl_Action_CreateUri 152
8.16.2. pl_Action_CreateGoto 153
8.16.3. pl_Action_CreateGotoR 153
8.16.4. pl_Action_CreateNamed 154
8.16.5. pl_Action_CreateJavaScript 154
8.16.6. pl_Action_CreateImpData 155
8.16.7. pl_Action_CreateLaunch 155
8.16.8. pl_Action_CreateSHField 156
8.16.9. pl_Action_CreateSubmitForm 156
8.16.10. pl_Action_CreateResetForm 157

8.16.11. pl_Action_CreateThread 157
8.16.12. pl_Action_AddAction 158
8.17. 対話フォームオブジェクト出力関数 158
8.17.1. pl_AcroForm_LoadTextField 160
8.17.2. pl_AcroForm_LoadRadioCheckButton 161
8.17.3. pl_AcroForm_LoadPushButton 161
8.17.4. pl_AcroForm_LoadChoiceField 162
8.17.5. pl_AcroForm_Set 163
8.18. エラー情報取得関数 163
8.18.1. エラーメッセージ取得関数 163

9. フォント設定 164

9.1. サポートされるフォント形式 164
9.2. フォント設定ファイル 164
9.2.1. 概要 164
9.2.2. フォント設定ファイルの要素・属性 164
9.3. Type1 フォント 166
9.3.1. フォントのファイル構成 166
9.3.2. 使用方法 166
9.3.3. フォントの名称指定 167
9.3.4. 文字コードとグリフの対応 168
9.3.5. 文字コードとグリフ名の対応付けの変更 169
9.3.6. 埋め込み 171
9.4. TrueType フォント、TrueType アウトラインを持つ OpenType フォント 172
9.4.1. 概要 172
9.4.2. 使用方法 172
9.4.3. 埋め込み 173
9.4.4. その他 173
9.5. PostScript アウトラインを持つ OpenType フォント 173
9.5.1. 概要 173
9.5.2. 使用方法 173

10. イメージ出力について 174

10.1. サポートされるラスターイメージフォーマット 174
10.2. イメージパススルー 175

11. サンプル 176

11.1. 呼び出し方法 176

12. 商標と著作権情報 177

12.1. 商標 177
12.2. 著作権情報 177

1. はじめに

本ライブラリは、PDF ファイル(PDF1.3～1.7 版)を作成する機能を提供するものである。Microsoft Windows Vista/7、Windows Server2003/2008/2008R2 上で動作する。

PDF1.3～PDF1.7 の仕様についてはそれぞれ、「PDF Reference,Second Edition」～「PDF Reference,Sixth Edition」を参照のこと。
本書内の PDF の用語については、原則的に上記 PDF Reference,Second Edition の邦訳版である株式会社ピアソンエデュケーション社刊「PDF リファレンス 第 2 版 Adobe Portable Document Format Version 1.3」（以下、PDF 仕様書)に従う。

1.1. コンパイラ

本ライブラリは Microsoft Visual Studio 2005 SP1 でビルドされている。
本ライブラリを利用する場合は、互換性のあるコンパイラを使用する事。
また、本ライブラリは「マイクロソフト セキュリティ情報 MS11-25」に対応しているため、
動作環境には以下の再頒布モジュールが必要となる。

・Microsoft Visual C++ 2005 Service Pack 1 再頒布可能パッケージ MFC のセキュリティ更新プログラム
（ http://www.microsoft.com/downloads/ja-jp/details.aspx?familyid=ae2e1a40-7b45-4fe9-a20f-2ed2923aca62&displaylang=ja）

2. 機能概要

2.1. テキスト

フォント、記述方向、位置を指定してテキストを出力することができる。設定されている条件に沿って、文字列の幅を計算して戻す機能を持つ。

1. サポートされるフォント
   本ライブラリがサポートするフォントは、以下のものである。
   (1) Type1 フォント
   Type1 フォントは、UNIX 環境では通常、拡張子 .afm と.pfb のファイルの組み合わせで、Windows 環境では .pfm と .pfb のファイルの組み合わせで使用される。このいずれの組み合わせもサポートする。
   (2) TrueType フォント
   TrueType フォントは拡張子として.TTF または .TTC を持つ TrueType フォントである。いずれもサポートする。
   (3) OpenType フォント
   OpenType フォントは、拡張子として、.TTF、.TTC、.OTF のいずれかを持つフォントである。いずれもサポートする。
2. フォントの埋め込みのサポート
   前項に記載した各フォントについて以下のように埋め込みをサポートする。

(1) Type1 フォントの埋め込み

.PFB ファイルが存在する場合、指定に従って埋め込みを行う。

(2) TrueType、OpenType フォントの埋め込み

TrueType、OpenType フォントには、フォントベンダによって埋め込みが制限されているものが存在する。このようなフォントを除いて、フォントの埋め込みをサポートする。使用した文字のグリフだけを埋め込むサブセット埋め込みを行う。

2.2. イメージ

イメージを PDF ファイルに出力することができる。出力時、BMP 形式のイメージを Flate、JPEG、JPEG2000 などで圧縮して出力すること、あるいは、必要な解像度にダウンサンプリングして出力することができる。JPEG、PNG、TIFF、GIF、JPEG2000 イメージについては、PDF ファイルに直接格納可能な形式であれば、ライブラリ内で加工を行わず、そのまま PDF に出力する。これをイメージパススルー方式と呼ぶ。これにより、イメージデータを高速に出力することができる。

本ライブラリへの出力前に、処理可能なイメージ形式か否かの判定を行うことができる。

2.3. グラフィックス

位置、種類、色、太さ、透明度などを指定して線(直線、曲線、円、四角形等)を出力することができる。色については DeviceGray、DeviceRGB、DeviceCMYK、あるいは ICC カラープロファイルを使用した色指定などの各種カラースペースをサポートする。

2.4. シェーディング

PDF のシェーディング機能が実現できる。例えば、ボールを表現する時、ボール表面の輝度と色彩によって、ボールの立体感を表現することができる。

2.5. 関数

シェーディング、あるいはスポットカラーの代替色などの計算に使用する関数オブジェクトをサポートする。

2.6. アウトライン

PDF のアウトライン（Acrobat 日本語版では「しおり」）の設定をサポートする。

2.7. 注釈

PDF には、各種の注釈機能が存在する。リンク注釈、テキスト注釈等をサポートする。

2.8. セキュリティ

Acrobat 4.0, Acrobat 5.0, Acrobat 6.0 相当のセキュリティ設定をサポートする。ユーザパスワード・マスタパスワードの設定、および、各種権限、暗号化方式などの設定が可能である。

2.9. PDF/X

PDF/X は ISO15930 で規定される、印刷データ交換用の PDF のサブセットである。本ライブラリでは、以下をサポートする。

ISO 15930-1:2001(PDF/X-1:2001,PDF/X-1a:2001)	PDF1.3 をベースとする
ISO 15930-3:2002(PDF/X-3:2002)	PDF1.3 をベースとする
ISO 15930-4:2003(PDF/X-1a:2003)	PDF1.4 をベースとする
ISO 15930-5:2003(PDF/X-2:2003)	PDF1.4 をベースとする
ISO 15930-6:2003(PDF/X-3:2003)	PDF1.4 をベースとする

2.10. PDF/A

PDF/A は ISO19005 で規定される、長期保存用の PDF のサブセットである。本ライブラリでは、以下をサポートする。

ISO 19005-1:2005	PDF1.4 をベースとする

ISO 19005-1 では、PDF/A に 2 種類の準拠レベル PDF/A-1a、PDF/A-1b が定義される。この双方をサポートする。

3. 構成ファイル

本ライブラリは以下のデータファイル および DLL から構成される。

3.1. フォント設定ファイル

フォントの格納される位置などを記述する xml ファイルである。このファイルは以下のように検索される。
環境変数"AH_FONT_CONFIGFILE" でこのファイルのフルパスを指定する。環境変数が定義されていない場合、本ライブラリが格納されるフォルダ内の font-config.xml をフォント設定ファイルとする。
このファイルが存在しない場合、Windows のシステムにインストールされているフォントが使用される。

3.2. ライブラリファイル

Windows 環境の場合のライブラリの構成ファイルを以下に示す。

PdlPDFCreator30.dll	: PDF 出力ライブラリ本体
PdlPDFRes30.dll	: CMap 情報取得ライブラリ
PdlPDFToolPage30.dll	: PDF Import 用ライブラリ
PdlCommon30.dll	: 共通ライブラリ
PdlDMC30.dll	: 共通ライブラリ
PdlFont30.dll	: フォントサービスモジュール
PdlGraphic30.dll	: グラフィックサービスモジュール
icudt40.dll、icuin40.dll、icuuc40.dll	: icu ライブラリ

4. 戻り値

エラーコードの一覧を以下に示す。

エラー名称	値	内容
PL_ERR_F_Font	1001	フォントの生成に失敗した
PL_ERR_F_GetCW	1002	文字幅の取得に失敗した
PL_ERR_F_GetTTIdx	1003	TrueType フォントのグリフインデクスの取得に失敗した
PL_ERR_F_GetT1CC	1004	Type1 フォントの CharCode の取得に失敗した
PL_ERR_F_QueryGlyExist	1005	グリフの有無の判定に失敗した
PL_ERR_F_GetTTDefCW	1006	デフォルト文字の幅の取得に失敗した
PL_ERR_F_T1AfmEncodeScheme	1007	AFM ファイルのエンコード情報の取得に失敗した
PL_ERR_F_GetTTFontSig	1008	TrueType フォントのサポートする文字集合の取得に失敗した
PL_ERR_F_GetOutlineTMtx	1009	フォントのアウトラインメトリクス情報の取得に失敗した
PL_ERR_F_GetT1EmbInfo	1010	Type1 フォントの埋め込み情報の取得に失敗した
PL_ERR_F_GetTTEmbInfo	1011	TrueType フォントの埋め込み情報の取得に失敗した
PL_ERR_F_GetPSName	1012	PostScript フォント名の取得に失敗した
PL_ERR_F_GetFFName	1013	フォントファミリーの名称取得に失敗した
PL_ERR_F_GetT1CCAsgnMap	1014	Type1 フォントに割り当てる情報の取得に失敗した
PL_ERR_F_SetGpNum	1015	フォントのグループ番号の設定に失敗した
PL_ERR_F_GetFType	1016	フォントタイプの取得に失敗した
PL_ERR_F_GetFStyle	1017	フォントスタイルの取得に失敗した
PL_ERR_F_QueryEmbStatus	1018	フォントの埋め込みプロパティの判定に失敗した
PL_ERR_F_GetCIDSysInfo	1019	OpentypeCID フォントの CID 情報の取得に失敗した
PL_ERR_F_T1PfmEncodeScheme	1020	Type1 pfm フォントのエンコーディング情報の取得に失敗した
PL_ERR_F_Lan	1021	フォント生成時にフォントの別名設定の誤りを検出した
PL_ERR_F_Node	1022	フォント生成時にフォント設定ファイルの誤記のため失敗した
PL_ERR_F_Weight	1023	フォント生成時に Weight 設定不正のため失敗した
PL_ERR_F_Alias	1024	フォント生成時にフォントの別名の重複定義のために失敗した
PL_ERR_F_InitFile	1025	フォント生成時にフォント設定ファイルの検索に失敗した
PL_ERR_F_FontFile	1026	フォント生成時にフォントファイルの検索に失敗した
PL_ERR_F_FontFolder	1027	フォント生成時にフォントフォルダの検索に失敗した
PL_ERR_F_StyleNotFound	1028	フォント生成時に指定されたスタイルのフォントの検索に失敗した
PL_ERR_F_FontFileRead	1029	フォント生成時に指定されたフォントファイルの読込みに失敗した
PL_ERR_G_ConvImgData	2001	イメージデータの変換に失敗した
PL_ERR_G_GetImgData	2002	イメージデータの取得に失敗した
PL_ERR_G_Load	2003	イメージのロードに失敗した
PL_ERR_G_UnSupType	2004	サポートしないイメージタイプである
PL_ERR_G_NoLZWLicense	2005	LZWLicense 禁止状態で、LZW の解凍が必要なイメージが入力された(これは現在、発生しない)
PL_ERR_G_ImgData	2006	WhitePoint あるいは BlackPoint の取得に失敗した
PL_ERR_IMG_InvalidHandle	2007	無効なイメージハンドルが指定された
PL_ERR_G_InvalidFrameNo	2008	無効なフレーム番号が指定された
PL_ERR_PDFX_Version	3001	マッチングしない PDF/X バージョン指定(これは現在、発生しない)
PL_ERR_PDFX_ColorSpace	3003	PDF/X 出力時に非対応のカラースペースが指定された
PL_ERR_PDFX_ImgColorSpace	3004	PDF/X 出力時に非対応のカラースペースの画像が出力された
PL_ERR_PDFX_ImgCompress	3005	PDF/X 出力時に非対応の圧縮方式の画像が出力された
PL_ERR_PDFX_ImgSMask	3006	PDF/X 出力時に alpha チャネル付きの画像が出力された
PL_ERR_PDFX_ImgMask	3007	PDF/X 出力時に Mask データが出力された
PL_ERR_PDFX_ColorProfile	3008	PDF/X 出力時に指定できない colorprofile が指定された
PL_ERR_PDFX_GState	3009	PDF/X 出力時に指定できない透明属性が指定された
PL_ERR_PDFX_ViewPref	3010	PDF/X 出力時に指定できない表示設定が設定された

PL_ERR_PDFX_Encrypt	3011	PDF/X 出力時に指定できないセキュリティ設定が指定された
PL_ERR_PDFX_AnnotPos	3012	PDF/X 出力時に指定できない Annotation 位置が指定された
PL_ERR_PDFX_AcroFormPos	3013	PDF/X 出力時に指定できない AcroForm 位置が指定された
PL_ERR_PDFX_Action	3014	PDF/X 出力時に指定できない Action が指定された
PL_ERR_PDFX_Info	3015	PDF/X 出力時に指定できない Info 情報が指定された
PL_ERR_PDFX_ImportedPdfX	3016	PDF/X 出力時にインポートできない PDF が指定された
PL_ERR_PDFX_NotEmb_License	3017	PDF/X 出力時に埋込み不可のフォントが使用された
PL_ERR_PDFX_NotEmb_Support	3018	PDF/X 出力時に埋込みがサポートされないフォントが使用された
PL_ERR_PDFX_NotEmb_Option	3019	PDF/X 出力時にフォント埋め込みが指定されなかった
PL_ERR_PDFX_MoreOPI	3020	PDF/X 出力時に複数の出力インテントが使用された
PL_ERR_PDFX_ImportOPI	3021	PDF/X 出力時に埋込 PDF が未承認の出力インテントを持っている
PL_ERR_TAG_StructElementCrossed	3101	TaggedPDF 出力時に Tag の不整合を検出した
PL_ERR_TAG_UnNecessaryInfo	3102	TaggedPDF 出力時に不要な情報が設定された
PL_ERR_TAG_StructElementNoExist	3103	TaggedPDF 出力時に存在しない要素が使用された
PL_ERR_TAG_StructElementEnded	3104	TaggedPDF 出力時に、終了した要素が使用された
PL_ERR_TAG_ActivatePseudo	3105	TaggedPDF 出力時にダミー要素をアクティベートした
PL_ERR_TAG_ImportedTaggedPDF	3106	TaggedPDF 出力時に TaggedPDF ファイルのインポートが指定された
PL_ERR_AF_Empty	3201	空の AcroForm が指定された
PL_ERR_AF_InvalidParent	3202	無効な親が設定された
PL_ERR_AF_UnSettedParent	3203	親が設定されていない
PL_ERR_AF_Format	3204	無効なフォーマットが指定された
PL_ERR_AF_InvalidAP	3205	不正な外観が指定された
PL_ERR_FileOpen	3301	ファイルのオープンに失敗した
PL_ERR_FileRead	3302	ファイルの読込みに失敗した
PL_ERR_StreamRead	3303	ストリームの読込みに失敗した
PL_ERR_FileWrite	3304	ファイルの書き出しに失敗した
PL_ERR_Ins_Path	3305	パスの描画コマンドでエラーが発生した
PL_ERR_StreamWrite	3306	ストリームの書き出しに失敗した
PL_ERR_Ins_ColorMode	3307	カラータイプ設定コマンドでエラー発生
PL_ERR_Ins_PaintMode	3308	塗りつぶしまたはパスを描画するコマンドでエラーが発生した
PL_ERR_Ins_gsMode	3309	グラフィック状態設定コマンドでエラー発生
PL_ERR_Ins_TextParas	3310	テキストの属性設定コマンドでエラーが発生した
PL_ERR_InvalidPara_Encoding	3311	無効なエンコードの指定
PL_ERR_InvalidPara_Langtype	3312	無効な言語タイプの指定
PL_ERR_InvalidPara_FontName	3313	無効なフォント名称の指定
PL_ERR_InvalidPara_Annot	3314	無効な注釈名の指定
PL_ERR_InvalidPageParas	3315	無効なページパラメーターの指定
PL_ERR_FileNameEmpty	3316	ファイル名が空であった
PL_ERR_MemoryNotEnough	3317	メモリ不足
PL_ERR_TwoSamePassWord	3318	ユーザパスワードとマスタパスワードに同じ値が設定された
PL_ERR_InvalidUserPassWord	3319	無効なユーザパスワードが指定された
PL_ERR_InvalidOwnerPassWord	3320	無効なマスタパスワードが指定された
PL_ERR_Compress	3321	圧縮に失敗
PL_ERR_UnEmbedable_NoLicense	3322	埋め込みが許可されていないフォントの埋め込みが指定された
PL_ERR_UnEmbedable_NotSupport	3323	ライセンス以外の原因によるフォントの埋め込みエラー（アウトラインデータがない、あるいは、埋め込みがサポートされていないフォント）
PL_ERR_EmptyPoint	3324	ポインタが設定されていない
PL_ERR_FindCMap	3325	CMap の検索に失敗した
PL_ERR_InvalidPdfVersion	3326	PDF のバージョン設定が不正
PL_ERR_InvalidOutline	3327	Local Outline を設定時に不正なページ番号が指定された
PL_ERR_DuplicateNames	3328	Names 設定時に重複エラーを検出した
PL_ERR_InvalidNames	3329	Names 設定時に不正なページ番号が指定された

PL_ERR_InvalidLocalLink	3330	カレントページ或は不正なページ番号で Local Link を設定する時エラー発生した。
PL_ERR_InvalidAnnot	3331	注釈設定の呼び出しシーケンスの誤り
PL_ERR_ImportedFile	3332	PDF ファイルのインポート時に埋め込まれるファイルに不具合を検出した
PL_ERR_ImportedPage	3333	PDF ファイルのインポート時に指定されたページが存在しない
PL_ERR_UnImportableFile_PSW	3334	セキュリティが設定されている PDF ファイルのインポートが指定された
PL_ERR_UnImportableFile_Ver	3335	バージョンが高い PDF ファイルのインポートが指定された
PL_ERR_UnImportableFile_LZW	3336	LZW ライセンスが禁止されている状態で、LZW の解凍が必要な PDF ファイルのインポートが指定された(現在、発生しない)
PL_ERR_ImportedRes	3337	PDF ファイルのインポート時に Resource の読込でエラーが発生した
PL_ERR_ImportedCont	3338	PDF ファイルのインポート時に Contents の読込でエラーが発生した
PL_ERR_UnImportableFile_Pat	3339	Pattern 定義内で PDF のインポートを実行しようとした
PL_ERR_CS_InvalidWhitePoint	3340	WhitePoint の指定値に誤りを検出した
PL_ERR_CS_InvalidBlackPoint	3341	BlackPoint の指定値に誤りを検出した
PL_ERR_CS_InvalidGamma	3342	Gamma の指定値に誤りを検出した
PL_ERR_CS_InvalidRange	3343	カラースペースの値の範囲指定に誤りを検出した
PL_ERR_CS_Empty	3344	カラースペースの指定に誤りを検出した
PL_ERR_Color_Empty	3345	カラーの指定に誤りを検出した
PL_ERR_CS_UnsupportedICCVer	3346	未サポートのバージョンのカラープロファイルが指定された
PL_ERR_CS_UnsupportedICCDevice	3347	PDF でサポートされないデバイス用のカラープロファイルが指定された
PL_ERR_CS_UnsupportedICCColor	3348	PDF でサポートされないカラースペース用のカラープロファイルが指定された
PL_ERR_CS_UnAllowedColorSpace	3349	指定箇所では使用できないカラースペースが指定された
PL_ERR_IMG_UnColorized	3350	着色できないイメージに着色が指定された
PL_ERR_InvalidFunctionValue	3351	関数の指定値に誤りを検出した
PL_ERR_CS_LoadColorProfile	3352	カラープロファイルのロードに失敗した
PL_ERR_CS_UnsupportedColorSpace	3353	未サポートのカラースペースが指定された
PL_ERR_Function_Empty	3354	関数インデクスの指定に誤りを検出した
PL_ERR_Shading_Empty	3355	シェーディングインデクスの指定に誤りを検出した
PL_ERR_Pattern_Empty	3356	パターンインデクスの指定に誤りを検出した
PL_ERR_CMS_Empty	3357	カラープロファイルインデクスの指定に誤りを検出した
PL_ERR_CS_InvalidICCColor	3358	ICC カラー値の指定に誤りを検出した
PL_ERR_Form_Empty	3359	フォームインデクスの指定に誤りを検出した
PL_ERR_Annot_InvalidIcon	3360	無効な注釈アイコンが指定された
PL_ERR_NameLen	3361	名前の長さが規定の長さを超えた
PL_ERR_ObjNumber	3362	オブジェクト番号超過
PL_ERR_StateDepth	3363	グラフィックス状態のスタックがオーバーフローした
PL_ERR_InvalidUGMap	3364	Unicode とグリフ番号の対応指定に不整合を検出した
PL_ERR_Shading_InvalidPara	3365	無効なシェーディングパラメータが指定された
PL_ERR_CS_InvalidSpotValue	3366	PKI 暗号化エラー
PL_ERR_PKIEncrypt	3367	インポートできない未サポートの暗号化が指定された
PL_ERR_UnImportableFile_ENCRYPT	3368	インポートできない PKI 情報が指定された
PL_ERR_UnImportableFile_PKI	3369	Unicode とグリフ番号の対応指定に不整合を検出した
PL_ERR_PDFA_GState	3400	PDF/A 出力時に無効なグラフィックス状態が指定された
PL_ERR_PDFA_Encrypt	3401	PDF/A 出力時に暗号化が指定された
PL_ERR_PDFA_ImgSMask	3402	PDF/A 出力時に無効なソフトマスクが指定された
PL_ERR_PDFA_AnnotType	3403	PDF/A 出力時に無効な注釈が指定された
PL_ERR_PDFA_AnnotCA	3404	PDF/A 出力時に注釈に無効な CA が指定された
PL_ERR_PDFA_AnnotPara	3405	PDF/A 出力時に注釈に無効なフラグが指定された

PL_ERR_PDFA_ActionType	3406	PDF/A 出力時に無効なアクションが指定された
PL_ERR_PDFA_ImportedPdfA	3407	PDF/A 出力時に PDFA ファイルのインポートが指定された
PL_ERR_PDFA_ImgCompress	3408	PDF/A 出力時に無効なイメージ圧縮方法が指定された
PL_ERR_PDFA_ImgMask	3409	PDF/A 出力時に無効なイメージマスク画像が指定された
PL_ERR_PDFA_ColorSpace	3410	PDF/A 出力時に無効なカラースペースが指定された
PL_ERR_PDFA_ColorError	3411	PDF/A 出力時に無効なカラー値が指定された
PL_ERR_PDFA_ImgColorSpace	3412	PDF/A 出力時に無効なカラースペースを持つ画像が指定された
PL_ERR_PDFA_NotEmb_License	3413	PDF/A 出力時に埋め込み不可能なフォントが指定された
PL_ERR_PDFA_NotEmb_Support	3414	PDF/A 出力時に埋め込みできないフォントが指定された
PL_ERR_PDFA_NotEmb_Option	3415	PDF/A 出力時に埋め込みが指定されていない
PL_ERR_PDFA_Type1CharSet	3416	PDF/A 出力時に Type1 フォントの文字名と文字コードの取得に失敗
PL_ERR_PDFA_TriggerEvent	3417	PDF/A 出力時に無効なトリガイベントが指定された
PL_ERR_PDFA_MissGlyProhibited	3418	PDF/A 出力時にグリフのない文字が指定された
PL_ERR_PDFA_MultipleCP	3419	PDF/A 出力時に複数のカラープロファイルが指定された
PL_ERR_PDFA_ImportedCP	3420	PDF/A 出力時に未承認のカラープロファイルを持つ PDF がインポート指定された
PL_ERR_PDFA_EmptyOutputIntent	3421	PDF/A 出力時に出力インテントがない
PL_ERR_PDFA_ValidRealNumber	3422	PDF/A出力時に-32767～32767 の範囲を超える実数が指定された
PL_ERR_PDFA_Transparency	3424	PDF/A 出力時に互換性のない透明度が指定された
PL_ERR_OPIEmpty	3500	出力インテントがない
PL_ERR_OPConditionIdenEmpty	3501	出力条件識別子がない
PL_ERR_CSConvert	3502	カラースペース変換エラー

5. 制限事項

以下に主な制限事項を記載する。

5.1. PDF バージョン

PDF1.２以下は現在、未サポートである。

5.2. フォント

ビットマップフォント、Type３フォントは未サポートである。また、TrueType、OpenType は Unicode cmap を持つものを対象とする（Symbolic フォントは埋め込み処理を行う）。

5.3. カラースペース

Indexed、DeviceN カラースペースは未サポートである。

5.4. イメージ

JPEG、PNG、TIFF、GIF、JPEG2000 を直接 PDF に出力する形式でサポートする。また、直接 PDF に出力できない圧縮形式などは、一旦解凍を行ってから出力する。

カラースペースの変換は現在サポートされない。また、JPEG2000 については JasPer ライブラリの機能に依存する。

5.5. その他

Optional Content などの出力は現在、未サポートである。

5.6. PDF/X に関する注意事項

本ライブラリでは、PDF/X で使用が禁止されている機能が指定された場合に、原則的に、それを補正するような処理は行わない。たとえば、PDF/X で使用が禁止されている透明度が指定された場合はエラーとなる。
PDF/X に関する主な注意事項を以下に記載する。

- 各 PDF/X 共通の注意事項
  - ✧ フォント

    PDF/X の指定とともに、全フォントの埋め込みを指定する必要がある。
  - ✧ ページ設定(TrimBox、ArtBox 等)

    TrimBox、ArtBox は必須であり、、ユーザに設定されない場合は、CropBox、BleedBox の値を取得して使用する。ユーザが設定する場合、BleedBox、CropBox に含まれるサイズでなければならない。これを越える場合、本ライブラリがそのサイズになるように調整する。
  - ✧ 透明属性：

    透明属性は使用できない。
  - ✧ ページ設定の表示(viewarea、viewclip、printarea、printclip 等)

MediaBox あるいは BleedBox のみである。

- ✧ PDF ファイルの埋め込み

  その PDF/X バージョン及び output intent は在り合わせの PDF ファイルを兼有する。

- ✧ PDF ファイルのバージョン

  PDF-1.4：透明属性、ページ設定の表示などの内容を判別する必要がある。

- ✧ Info 中の Creator と Title:

  文書情報(Info 辞書)中の Creator と Title を設定しなければならない

- ✧ Action と JavaScripts

  PDF/X では Action と JavaScripts は使用できない。

- ✧ ExtGSState

  PDF/X では、ExtGSState の CA、ca は 1.0 でなければならない。また、BM は Normal でなければならない。

- PDF/X-1 の注意事項
  - ✧ カラースペース

    RGB、ICCBased、Lab などの ColorSpace を設定することはできない。

  - ✧ デフォルトカラースペース：

    Default ColorSpace を設定することはできない。

  - ✧ 画像

    RGB、Lab、ICCBased ColorSpace の画像を出力することはできない。

    Lzw 圧縮方法を使用する画像を出力することはできない。例えば、Gif、Lzw の Tiff など。

  - ✧ 出力デバイス

    Monochrome または Cmyk のでなければならない。

  - ✧ セキュリティ情報

    ユーザパスワード、及び印刷禁止の権限設定を指定することはできない。

- PDF/X-3 の注意事項
  - ✧ デフォルトカラースペース

    Default ColorSpace を設定しなければならない場合がある。例えば、RGB colorspace を使用する場合に、出力デバイスは RGB のではない。その時に、DefaultRGB ColorSpace を設定しなければならない。

  - ✧ セキュリティ情報

    ユーザパスワード、マスタパスワード、及びすべての権限設定も指定することはできない。

5.7. PDF/A に関する注意事項

PDF/A についても、PDF/A 同様、使用が禁止されている機能が指定された場合に、原則的に、それを補正するような処理は行わない。

6. 共通データ形式

次章以降の各種関数の説明内で共通に使用される構造体を記載する。

6.1. PL_PointTD

点を定義する構造体である。PL_PointTD 構造体の定義を以下に示す。

```
typedef struct {
    float    x,y;              // 点(x,y)の座標値
} PL_PointTD;
```

6.2. PL_RectangleTD

矩形を定義する構造体である。PL_RectangleTD 構造体の定義を以下に示す。

```
typedef struct {
    float    x1,y1,x2,y2;      // 矩形枠の左下角の点(x1,y1)及び右上角の点(x2,y2)の座標
} PL_RectangleTD;
```

6.3. PL_LinePatternTD

線のパターンを格納する構造体である。PL_LinePatternTD 構造体の定義を以下に示す。

```
typedef struct {
    int      n;                // 使用するパターンの数
    float*   lpPattern;        // パターン格納領域へのポインタ
    float    phrase;           // パターンフレーズ
} PL_LinePatternTD;
```

説明：

n：線パターンの段数

lpPattern：線のパターンデータへのポインタ

phrase：ラインの開始のフレーズ

pattern 配列に破線の一周期の破線パターンを指定する。パターン、およびフレーズの単位はユーザ空間上の単位である。実線の場合、 n=0、phrase=0 とする。以下に例を示す。

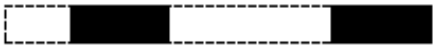
破線パターン	配列とフレーズ	説明
	[ ] 0	実線
	[3] 0	3 単位オン、3 単位オフ、・・・
	[2] 1	1 単位オン、2 単位オフ、2 単位オン、2 単位オフ、・・・
	[2 1] 0	2 単位オン、1 単位オフ、2 単位オン、1 単位オフ、・・・
	[3 5] 6	2 単位オン、3 単位オフ、5 単位オフ、3 単位オン、5 単位オフ・・・
	[2 3] 11	1 単位オン、3 単位オフ、2 単位オン、3 単位オフ、2 単位オン、・・・

表 6.3 破線パターンの例

[2 3] 11 の場合、n=2、phrase=11、pattern[0]=2、pattern[1]=3 を設定する。

6.4. PL_LineTD

線のプロパティ（例えば文字の取消し線、下線、上線など）を定義する構造体である。PL_LineTD の定義を以下に示す。

```
typedef struct {
    PL_LineTypeE    type;    // 線種
    float           width;   // 線幅
    PL_ColorTD      color;   // 線色
} PL_LineTD;
```

説明：

type：線のタイプ。以下に定義を示す。

```
typedef enum{
    PL_LT_SINGLE=1,          // 細かい実線
    PL_LT_DOUBLE,            // 二重実線
    PL_LT_DOT,               // 点線
    PL_LT_THICKDOT,          // 太い点線
    PL_LT_DASH,              // 破線
    PL_LT_THICKDASH,         // 太い破線
    PL_LT_DOTDASH,           // 鎖線
    PL_LT_THICKDOTDASH,      // 太い鎖線
    PL_LT_WAVE,              // 波線
    PL_LT_THICK              // 太い実線
} PL_LineTypeE;
```

width：線幅(ユーザ空間単位)

color:線の色(6.7 の説明参照)

6.5. PL_BorderTD

境界線の特性を示す構造体である。PL_BorderTD の定義を以下に示す。

```
typedef struct {
    PL_BorderSubtypeE    Subtype; // 境界線の線種
    float                W;       // 幅(ポイント単位)
    PL_LinePatternTD     Dash;    // 破線パターン(後述)
                                  // Subtype に破線(PL__BD_D)が指定された時のみ有効)
} PL_BorderTD;
```

説明：

Subtype は境界線の線種を指定する。定義を以下に示す。

```
typedef enum{
    PL_BD_S=0,      // S (実線)
    PL_BD_D,        // D (破線) 形状は Dash で指定する
    PL_BD_B,        // B (ベベル)
    PL_BD_I,        // I (インセット)
    PL_BD_U,        // U (下線)
    PL_BD_MAXNUM
}PL_BorderSubtypeE;
```

6.6. PL_ColorSpaceTD

カラースペースを示す構造体である。PL_ColorSpaceTD の定義を以下に示す。

```
typedef struct {
    PL_ColorTypeE           type;           // カラースペースのタイプ(下記参照)
    PL_UnColorPattern1TD    unColorPat1;    // 色無しパターン用パラメータ(下記参照)
    PL_CalGrayTD            calGray;        // CalGray カラースペース用パラメータ
    PL_CalRGBTD             calRGB;         // CalRGB カラースペース用パラメータ
    PL_LabTD                lab;            // Lab カラースペース用パラメータ
    PL_ICCBasedTD           iccBased;       // ICCBased カラースペース用パラメータ
    PL_SpotTD               spot;           // Separation カラースペース用パラメータ
} PL_ColorSpaceTD;
```

- カラースペース定義

```
typedef enum {
// デバイスカラースペース(DeviceRGB,DeviceGray,DeviceCMYK)
    PL_CS_DEVICERGB = -100, PL_CS_DEVICEGRAY, PL_CS_DEVICECMYK,
// 特殊カラースペース(Pattern,Separation)
    PL_CS_PATTERN, PL_CS_SPOT,
// CIE-Based カラースペース(CalGray,CalRGB,Lab,ICCBased)
    PL_CS_CALGRAY, PL_CS_CALRGB, PL_CS_LAB, PL_CS_ICCBASED
    PL_CS_NONE   = 0,                   // no color,transparent
} PL_ColorTypeE;
```

- 色無しパターンカラースペース用パラメータ定義

```
typedef struct {
    PL_ColorTypeE   csType;     // 色無しパターン用カラースペース
```

```
    int                 csIdx;      // カラースペースインデクス(csType が
                                    // デバイスカラースペースでない場合のみ指定のこと)
} PL_UnColorPattern1TD;
```

- CalGray カラースペース用パラメータ定義

```
typedef struct {
    float whitePoint[3]; // CIE1931XYZ 空間のホワイトポイントの 3 刺激値 [Xw Yw Zw]
                         // (指定必須：Xw と Zw は正の数、Yw は 1.0 固定)
    float blackPoint[3];  // CIE1931XYZ 空間のブラックポイントの 3 刺激値 [Xb Yb Zb]
                         // (3 値とも 0 以上の数。デフォルト値：[0 0 0] )
    float gamma;         //グレイ成分のガンマ値(正の値、通常 1 以上。デフォルト値：1.0)
} PL_CalGrayTD;
```

- CalRGB カラースペース用パラメータ定義

```
typedef struct {
    float whitePoint[3];        // CIE1931XYZ 空間のホワイトポイントの 3 刺激値 [Xw Yw Zw]
                                // (指定必須：Xw と Zw は正の数、Yw は 1.0 固定)
    float blackPoint[3];        // CIE1931XYZ 空間のブラックポイントの 3 刺激値 [Xb Yb Zb]
                                // (3 値とも 0 以上の数。デフォルト値：[0 0 0] )
    float gamma[3];             //カラースペースの 3 成分 R,G,B 用のガンマ値 [Gr Gg Gb]
                                // (デフォルト値：[1 1 1])gamma
    float matrix[9];            //最終 XYZ のデコードされた成分 A,B,の線形解釈を指定する配列
                                // [Xa Ya Za Xb Yb Zb Xc Yc Zc] (デフォルト値：[1 0 0 0 1 0 0 0 1])
} PL_CalRGBTD;
```

- Lab カラースペース用パラメータ定義

```
typedef struct {
    PL_LabStdE  labStd;         //下記参照(PL_LAB_D50 指定時、以下の設定は不要)
    float       whitePoint[3    // CIE1931XYZ 空間のホワイトポイントの 3 刺激値 [Xw Yw Zw]
                                // (指定必須：Xw と Zw は正の数、Yw は 1.0 固定)
    float       blackPoint[3]; // CIE1931XYZ 空間のブラックポイントの 3 刺激値 [Xb Yb Zb]
                                // (3 値とも正の数)
    float       range[4];       // a*,b*の有効範囲を示す 4 値 [amin amax bmin bmax]
                                //   amin<=a*<=amax かつ bmin<=b*<=bmax
} PL_LabTD;
typedef enum {
    PL_LAB_NONE = 0,
    PL_LAB_D50   = 1            // D50 光源を使用する:WhitePoint [0.9462 1.0 0.8249],
```

```
                                    // BlackPoint([0 0 0]),Range [-128 127 -128 127]
} PL_LabStdE;
```

- ICCBased カラースペース用パラメータ定義

```
typedef struct {
    PL_CPHandle hColorProfile;          // pl_LoadColorProfile()で取得したハンドル
} PL_ICCBasedTD;
```

- セパレーションカラースペース用パラメータ定義

```
typedef struct {
    UCHAR_t*        lpColorName;        //カラー名
    PL_ColorTypeE alternateCSType;      //代替カラースペース
    int             alternateCSIdx;     //代替カラースペース用パラメータ定義
                                        // alternateeCSType がデバイスカラースペースで無い場合のみ必要
    float           alternateColor[4];  // 代替カラー値
                                        // alternateCSType が示すカラースペースの成分数だけ使用
    int             transFuncIdx;       // 色調変換関数のインデクス(0 であれば自動設定)
} PL_SpotTD;
```

説明：

- PDF には、パターン定義内では色を指定せずに、使用時にパターンの外部で 1 色を指定して使用するタイプの色無しパターンと呼ばれるものが存在する。unColorPat1 は、この場合のみ使用する。unColorPat1 の csType にその(1 色の)カラースペース(PL_CS_PATTERN は指定不可)を指定する。この csType がデバイスカラースペースでない場合、各カラースペースのパラメータ定義が必要となる。この場合 pl_LoadColorSpace で事前に使用するカラースペースをロードしておき、ここで取得したカラースペースインデクスを csIdx に設定する必要がある。
- Lab カラースペースで D50 光源を使用する場合、フラグを設定することでその他のパラメータを自動設定とすることができる。
- セパレーションカラースペースでは、代替色を指定する必要がある。また、この代替色とカラー値の変換関数を transFuncIdx に指定する必要がある。transFuncIdx が 0 の場合、指数補間を行う関数を自動で割り当てる。

  また、lpColorName におてい All と None は PDF では特別な意味を持つ (All はすべての色版で表示される。None はどの色版でも表示されない)。

  この場合も、他の引数については通常のセパレーションカラースペースの場合と同様に、正しい値を指定する必要がある。

6.7. PL_ColorTD

色の値を格納する構造体である。PL_ColorTD の定義を以下に示す。

```
typedef struct {
    PL_ColorTypeE   CSMode;  //カラースペース
    float           a,b,c,d;    //カラー値;
} PL_ColorTD;
```

説明：

1.CSMode：カラースペースを指定する。

2.a,b,c,d：各カラースペースのカラー値

- PL_CS_DEVICEGRAY：a に Gray 値を設定する(その他は設定不要。Gray 値は(0.0～1.0)の範囲)
- PL_CS_DEVICERGB：a に Red 値、b に Green 値、c に Blue 値を設定する(d 値は設定不要、Red,Green,Blue 値は 0.0～1.0 の範囲)
- PL_CS_DEVICECMYK：a に Cyan 値、b に Magenta 値、c に Yellow 値、d に Black 値を設定する(Cyan,Magenta,Yellow,Black 値は 0.0～1.0 の範囲)
- PL_CS_PATTERN：色無しパターンの場合、カラースペースに従った値を設定する
- PL_CS_SPOT：a にインクの量を設定する(0.0～1.0 の範囲)
- PL_CS_CALGRAY：a に Gray 値を設定する(0.0～1.0 の範囲)
- PL_CS_CALRGB：a に Red 値、b に Green 値、c に Blue 値を設定する(0.0～1.0 の範囲)
- PL_CS_LAB：a に L*値、b に a*値、c に b*値を設定する。L*値は 0～100 の範囲、a*,b*値はカラースペース定義時に指定した range の範囲とする。
- PL_CS_ICCBASED：カラープロファイルの示すカラースペースに沿った値を設定する。

6.8. PL_DestTD

リンク注釈、アウトライン(しおり)などで使用されるリンク先を設定する構造体である。DestTD の定義を以下に示す。

```
typedef struct {
    PL_DestStyleE   flag;           // フラグ
    float           a1,a2,a3,a4;    // 表示方法の指定情報（flag の説明参照）
    int             PgNo;           // ページ番号（flag の説明を参照）
} PL_DestTD;
```

説明：

PL_DestStyleE の定義を以下に示す

```
typedef enum{
        PL_DEST_XYZ=1       // a1:left、a2：top、a3：zoom を指定する
            // 座標(left、top)をウィンドウの左上角に置き、且つ zoom でページを拡大して表示する
            // (1.0 で 100%)。 zoom が NULL の場合、そのままカレント値が使用される。
        PL_DEST_FIT,
            // 水平方向、垂直方向ともにウィンドウ内に
```

// 全体が収まる倍率で表示する。

PL_DEST_FITH, // a1：top を指定する

// 垂直座標 top をウィンドウの上部に置き、ページ幅全体がウィンドウに収まる倍率で

// 表示する。

PL_DEST_FITV, //a1：left を指定する

// 水平座標 left をウィンドウの左端に置き、ページの高さ全体がウィンドウ内に

// 収まる倍率で表示する。

PL_DEST_FITR, //a1：left、a2：bottom、a3：right、a4：top を指定する

// 座標 left、bottom、right、top で指定する矩形が、水平方向、垂直方向共に

// ウィン/ドウ内に収まる倍率で表示する。

PL_DEST_FITB,

// 境界ボックス全体が水平方向、垂直方向共にウィンドウ内に収まる倍率で表示する。

PL_DEST_FITBH , //a1:top を指定する

// 垂直座標 top をウィンドウの上端に置き、ページの境界ボックスの全体がウィンドウ

// 内に収まる倍率で表示する。

PL_DEST_FITBV, //a1,:left を指定する

// 水平座標 left をウィンドウの左端に置き、ページの境界ボックスの高さがウィンドウ

// 内に納まる倍率で表示する。

} PL_DestStyleE;

なお、a1、a2、a3、a4 に NULL を設定する場合、マクロ PL_DESTNULL が定義されるのでこれを使用する。

✧ PgNo : Outline の宛先のページ番号を設定する。1 オリジンである。0 は、カレントページを指定するものとする。現在、リンク注釈ではこの構造体の PgNo は参照されない。

6.9. UCHAR_t

1. これは本ライブラリで定義される Unicode のデータ型である。
2. 本ライブラリの Unicode インタフェースで使用される。プラットフォームによって、バイト数も違っているため、注意が必要である。
3. 本ライブラリで用いる Unicode インタフェースはすべてシステムエンディアンと同じバイトオーダーの Unicode である。

7. 関数一覧

本ライブラリの関数一覧を以下に示す。

分類	関数名	
基本	**pl_Initial,**	**pl_Finish**
	pl_OpenPdf,	**pl_ClosePdf,**
	pl_InitFonts,	**pl_FinishFonts,**
	pl_Abort	**pl_OpenPdf_Stream**
	pl_ClosePdf_Temp	**pl_ReOpenPdf**
オプション設定　一般	**pl_SetPdfVersion,**	**pl_PutEncryptOpt,**
	pl_PutComprssOpt2,	**pl_PutASCIIFilter,**
	pl_SetObjOpt,	**pl_SetDocInfo,**
	pl_SetBaseUri,	**pl_PutPageModeOpt,**
	pl_PutPageLayoutOpt,	**pl_PutViewPreferences,**
	pl_PutOpenAction,	**pl_PutPageLabelOpt,**
	pl_SetNames,	
	pl_SetDocumentLang,	**pl_SetTaggedPdfMode**
	pl_SetTaggedPdfOpt,	**pl_PutOpenActionByName**
	pl_PutEmbOpt,	**pl_PutMissGlyphOpt,**
テキスト	**pl_PutEncodeOpt,**	
	pl_SetImgCompOpt pl_PutDownSampling,	
イメージ		**pl_SetTransImgProcMode,**
	pl_SetNoPassOpt,	**pl_SetNoBitConvOpt**
	pl_SetImageColorProfileOpt,	
PDF/X	**pl_SetPdfXOpt,**	**pl_SetPdfXCSConvOpt,**
PDF/A	**pl_SetPdfAOpt,**	**pl_SetPdfACSConvOpt**
ページ制御	**pl_CreatePages,**	**pl_ClosePages,**
	pl_CreatePage,	**pl_ClosePage,**
	pl_SetPagePaper,	**pl_SetCropOffset,**
	pl_SetBleedOffset,	
	pl_SetTrimOffset,	**pl_SetArtOffset**
グラフィックス状態設定	**pl_PushState,**	**pl_PopState,**
	pl_SetCm,	**pl_SetLineWidth,**
	pl_SetLineCap,	**pl_SetLineJoin,**
	pl_SetMiterLimit,	**pl_SetLineType,**
	pl_SetIntent,	**pl_SetGraphState,**
	pl_SetColor,	**pl_LoadColorSpace,**
	pl_SetColorSpace,	**pl_LoadColorProfile,**
	pl_SetDefaultColorSpace,	**pl_LoadColorProfile_Stream,**
	pl_LoadColorProfile_Buffer,	
	pl_CheckColorProfile,	**pl_CheckColorProfile_Stream,**
	pl_GetColorProfileInformation	
パス関連オペレータ　パス構築	**pl_GraphicTo,**	**pl_LineTo,**
	pl_CurveTo,	**pl_CurveTo_v,**
	pl_CurveTo_y,	**pl_Circle,**
	pl_ClosePath,	**pl_Rect,**
パスペイント	**pl_Paint,**	
クリッピングパス	**pl_PathClip,**	
テキストオペレータ	**pl_SetFont,**	**pl_SetWriteDirect,**
	pl_SetTm,	**pl_SetTextParas,**
	pl_TextTo,	**pl_NextLine,**
	pl_ShowTextU,	**pl_ShowTextU2,**
	pl_ShowTextGI,	**pl_GetCharWidth,**
	pl_GetStrWidthU,	**pl_GetStrWidthU2,**
	pl_GetMissGlyphChar,	**pl_GetCurAssignedFont,**

イメージ操作	**pl_CheckImage, pl_CheckImageMask** **pl_InitImage, pl_InitImageMask,** **pl_OpenImage, pl_OpenImage_Stream** **pl_OpenImageWithMask, pl_OpenImageWithMask_Stream** **pl_PutImage, pl_CloseImage,** **pl_FinishImage,** **pl_ColorizeImage,** **pl_GetImageWH, pl_GetImageWH_Stream** **pl_GetImageColorProfile, pl_GetImageColorProfileF** **pl_GetImageColorProfile_Stream** **pl_CheckImage_Stream, pl_CheckImageMask_Stream,** **pl_LoadImage_Stream, pl_LoadImageMask_Stream,** **pl_LoadImageWithMask_Stream, pl_PutImageByHandle,** **pl_RemoveImage_Stream, pl_GetImageColorProfileByHandle,** **pl_GetImageColorProfileFByHandle_Stream,** **pl_GetImageWHByHandle, pl_ColorizeImageByHandle**
関数オブジェクト出力	**pl_InitFunction, pl_CreateFunction,** **pl_FreeFunction**
パターン定義	**pl_BgnPattern1, pl_EndPattern1,** **pl_CreateShading, pl_CreatePattern2,** **pl_SetPattern, pl_Shading**
アウトライン出力	**pl_AddOutlineLevel, pl_AddOutlineLevel_Local,** **pl_AddOutlineLevel_Remote,** **pl_CloseOutlineLevel,** **pl_AddOutline, pl_AddOutline_Local,** **pl_AddOutline_Remote**
注釈オブジェクト出力	**pl_Annot_Bgn, pl_Annot_SetCommData,** **pl_Annot_End,** **pl_Annot_SetText, pl_Annot_SetFText,** **pl_Annot_SetLine, pl_Annot_SetSquare,** **pl_Annot_SetCirle, pl_Annot_SetPolygon,** **pl_Annot_SetPolyLine ,** **pl_Annot_SetHighlight, pl_Annot_SetUnderline,** **pl_Annot_SetSquiggly, pl_Annot_SetStrike,** **pl_Annot_SetStamp, pl_Annot_SetCaret ,** **pl_Annot_SetInk, pl_Annot_SetAttFile,** **pl_Annot_SetSound,** **pl_Annot_SetPopUp, pl_Annot_SetMovie** **pl_Annot_SetLinkSrc, pl_Annot_SetLinkDest,** **pl_Annot_SetLocalLink, pl_Annot_SetLinkOther,** **pl_Annot_SetRemoteLink,**
PDF インポート	**pl_ImpPdf_Initial, pl_ImpPdf_Check,** **pl_ImpPdf_GetPgCount, pl_ImpPdf_GetVer,** **pl_ImpPdf_GetSecurity, pl_ImpPdf_SetPgNo,** **pl_ImpPdf_GetPgSize, pl_ImpPdf_GetPgRotate,** **pl_ImpPdf_Do, pl_ImpPdf_Finish,** **pl_ImpPdf_GetPgBox, pl_ImpPdf_SetBoxMode,** **pl_ImpPdf_DoEx, pl_ImpPdf_IsTaggedPDF** **pl_ImpPdf_GetOutputIntent, pl_ImpPdf_GetOutputIntentCount,** **pl_ImpPdf_Initial_Stream**
FormXObject 設定	**pl_BgnForm, pl_EndForm** **pl_PutForm**
PDF/X 操作	**pl_LoadStdOutputIntent, pl_LoadGenOutputIntent,** **pl_SetOutputIntent**
TaggedPDF 操作	**pl_LoadLayoutAttr_BLSE, pl_LoadLayoutAttr_ILSE** **pl_LoadLayoutAttr_ILLUE, pl_LoadListAttr** **pl_LoadTblAttr, pl_BgnMKConternt**

	pl_EndMKContent, pl_ActivateMKElement **pl_BgnMKContentLeaf, pl_EndMKContentLeaf**
Action 作成	**pl_Action_CreateUri, pl_Action_CreateGoto,** **pl_Action_CreateGotoR, pl_Action_CreateNamed,** **pl_Action_CreateJavaScript, pl_Action_CreateImpData,** **pl_Action_CreateLaunch, pl_Action_CreateSHField,** **pl_Action_CreateSubmitForm, pl_Action_CreateResetForm,** **pl_Action_CreateThread, pl_Action_AddAction,**
対話フォーム	**pl_AcroForm_LoadTextField, pl_AcroForm_LoadPushButton,** **pl_AcroForm_LoadRadioCheckButton,** **pl_AcroForm_LoadChoiceField,** **pl_AcroForm_Set**
エラー情報	**pl_GetErrorMessage**

注意事項

- 以下の各関数において、PDF1.4 と記載されている項目は、PDF1.3 指定時は無視される。同様に、PDF のバージョンが記載されている項目はそれ未満のバージョン出力時は無視される。

8. 関数説明

8.1. 基本関数

8.1.1. 初期化関数

```
PDFAPI PL_ERROR pl_Initial(hPDF* pctp,
                    PL_FLOATPRECISION floatPrecision = PF_FLOATPRECISION_DEFAULT,)
```

機能：

初期化関数。必要なメモリを獲得する。この関数は本ライブラリを使用する場合に、最初に呼び出す必要がある。

引数：

pctp：PDF ファイルのポインタのポインタ

floatPrecision:浮動小数点数を処理する制度を指定する。指定された小数点以下の桁数を出力する。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.1.2. 終了関数

```
PDFAPI PL_ERROR pl_Finish (hPDF* pctp)
```

機能：

終了関数。メモリの開放などを行う。本ライブラリを使用する場合、最後に、呼び出す必要がある。

引数：

pctp ： PDF ファイルのポインタ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.1.3. 強制終了関数

```
PDFAPI void pl_Abort (hPDF* pctp)
```

機能：

強制終了関数。メモリの開放などを行う。PDF 出力中に、上位から処理を打ち切る場合に使用する。

引数：

pctp ： PDF ファイルのポインタ

戻り値：

なし

8.1.4. PDF ファイルオープン関数

```
PDFAPI PL_ERROR pl_OpenPdf(hPDF ctlp, const char* fn, PL_CoordTypeE coord,float pw,float ph)
```

機能：

出力する PDF ファイルのファイルパスを指定する。

引数：

ctlp： PDF ファイルのポインタ

fn：PDF ファイル名

coord：座標系設定オプション

```
typedef enum {
        PL_CT_PDF = 0,  // PDF ファイル形式の座標系の使用を指定する。
                // 原点をページの左下隅にもち、X 軸正方向を右、Y 軸正方向を下とする座標
        PL_CT_RTF = 1   // RTF ファイル形式の座標系の使用を指定する。
                // 原点をページの左上隅にもち、X 軸正方向を右、Y 軸正方向を下とする座標
} PL_CoordTypeE;
```

float pw,float ph：それぞれ、デフォルトの用紙の幅と高さとなる。単位はデフォルトユーザ空間の単位である 1/72 インチである。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.1.5. PDF ファイルオープン関数(ストリーム)

PDFAPI PL_ERROR pl_OpenPdf_Stream(hPDF ctlp,std::ostream& osmPdf,PL_CoordTypeE coord,float pw,float ph)

機能：

出力に用いる PDF ストリームを指定する。

引数：

ctlp： PDF ファイルのポインタ

osmPdf：PDF 出力ストリーム

coord：座標系設定オプション

float pw,float ph：それぞれ、デフォルトの用紙の幅と高さとなる。単位はデフォルトユーザ空間の単位である 1/72 インチである。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.1.6. PDF クローズ関数

PDFAPI PL_ERROR pl_ClosePdf(hPDF ctlp)

機能：

PDF ファイル、またはストリームを閉じる

引数：

ctlp ： PDF ファイルのポインタ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.1.7. フォントデータ初期化関数

PDFAPI PL_ERROR pl_InitFonts(hPDF ctlp)

機能：

フォントデータを初期化し、フォント埋め込み情報を設定する。

引数：

ctlp ： PDF ファイルのポインタ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.1.8. フォントデータ終了関数

PDFAPI PL_ERROR pl_FinishFonts(hPDF ctlp)

機能：

フォント管理に使用したメモリを開放する。pl_ClosePdf()関数でフォント情報を使用するため、この関数は pl_ClosePdf()関数のあとに呼び出すこと。

引数：

ctlp ： PDF ファイルのポインタ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.1.9. PDF ファイル一時クローズ関数

PDFAPI PL_ERROR pl_ClosePdf_Temp(hPDF ctlp)

機能：

一時的にファイルをクローズする。

引数：

ctlp ： PDF ファイルのポインタ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

分冊出力時、多数のファイルを同時に開いたまま処理し、最後にリンクの宛先などを指定してファイルを閉じる。この場合、分冊ファイルの数が多くなると、別のファイルが開けない場合がある。このような場合に、本関数により、処理中のファイルを一旦閉じることができる。

最終の pl_ClosePages を呼び出した後、本関数によりファイルを閉じ、分冊処理内でのリンク先などが決定した後、再オープン関数を呼出し後、リンク注釈、アウトラインの宛先を設定し、クローズ関数を呼び出す。

8.1.10. PDF ファイル再オープン関数

PDFAPI PL_ERROR pl_ReOpenPdf(hPDF ctlp)

機能：

PDF ファイル一時クローズ関数で閉じられている PDF ファイルを再度オープンする。

引数：

ctlp ： PDF ファイルのポインタ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.2. オプション設定関数

出力する PDF ファイルの各種設定を行う関数である。

オプション設定関数の呼び出しタイミングについて

pl_SetPdfVersion、pl_SetDocInfo、pl_putEncryptOpt で設定されるオプションの内容は、pl_OpenPdf 関数で使用される。このため、これらの関数は、pl_Initial 関数呼出し後、pl_OpenPdf 関数の呼び出し前に行なう必要がある。

pl_PutCompressOpt2、pl_PutASCIIFilter、pl_SetObjOpt、pl_PutDownSampling、 pl_PutEmbOpt、pl_PutEncodeOpt、pl_PutMissGlyphOpt で設定されるオプションの内容は pl_InitFonts 関数で参照される。このため、これらの関数は、pl_Initial 関数呼出し後でもよいが、pl_InitFonts 関数の呼び出し前には行なう必要がある。

pl_SetTransImgProcMode、 及び pl_SetImgCompOpt で設定されるオプションの内容は Image 関数で参照される。このため、これらの関数は、pl_Initial 関数呼出し後でもよいが、Image 関数の呼び出し前には行なう必要がある。

上記以外のオプション設定関数は、pl_Initial 関数から、pl_ClosePdf 関数間で任意に使用可能である。

8.2.1. PDF バージョン設定関数

PDFAPI PL_ERROR pl_SetPdfVersion(hPDF ctlp,PL_Version PDFVer)

機能：

出力する PDF 文書のバージョンを設定する。現在、PDF1.3、PDF1.4、PDF1.5. および PDF1.6 をサポートする。デフォルトは PDF1.4 である。

引数：

ctlp ： PDF 文書のポインタ

PDFVer ：PDF 文書のバージョンオプション

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_Version の定義を以下に示す。

```
typedef enum
{
        PL_VER10 = 0,
        PL_VER11 = 1,
        PL_VER12 = 2,
        PL_VER13 = 3,
        PL_VER14 = 4,
        PL_VER15 = 5,
        PL_VER16 = 6,
        PL_VER17 = 7,
        PL_VERMAX = PL_VER17,
        PL_VERMIN = PL_VER13,
        PL_OTHER = 0xFF
        PL_PDFX_1_2001 = 0x0100 | PL_VER13, //base on pdf1.3
        PL_PDFX_1a_2001 = 0x0200 | PL_VER13, //base on pdf1.3
        PL_PDFX_3_2002 = 0x0300 | PL_VER13, //base on pdf1.3
        PL_PDFX_1a_2003 = 0x0400 | PL_VER14, //base on pdf1.4
        PL_PDFX_2_2003 = 0x0500 | PL_VER14, //base on pdf1.4
        PL_PDFX_3_2003 = 0x0600 | PL_VER14, //base on pdf1.4
        PL_PDFX_OTHER = 0x0F00,
        PL_PDFA_1a_2005 = 0x1000 | PL_VER14,
        PL_PDFA_1b_2005 = 0x2000 | PL_VER14,
        PL_PDFA_OTHER = 0xF000,
        PL_PDFX_1a_2003_PDFA_1a_2005= PL_PDFX_1a_2003 | PL_PDFA_1a_2005,
        PL_PDFX_2_2003_PDFA_1a_2005 = PL_PDFX_2_2003   | PL_PDFA_1a_2005,
        PL_PDFX_3_2003_PDFA_1a_2005 = PL_PDFX_3_2003   | PL_PDFA_1a_2005,
        PL_PDFX_1a_2003_PDFA_1b_2005= PL_PDFX_1a_2003 | PL_PDFA_1b_2005,
        PL_PDFX_2_2003_PDFA_1b_2005 = PL_PDFX_2_2003   | PL_PDFA_1b_2005,
        PL_PDFX_3_2003_PDFA_1b_2005 = PL_PDFX_3_2003   | PL_PDFA_1b_2005
} PL_Version;
```

PL_PDFX_1_2001 から PL_PDFX_3_2003 までは PDF 文書の PDF/X バージョンオプション

8.2.2. セキュリティ設定関数

PDFAPI PL_ERROR pl_PutEncryptOpt(hPDF ctlp, PL_EncryptOptionTD * lpSetEncryptOpt)

機能：

セキュリティオプションを設定する。設定可能な内容は出力する PDF のバージョンによって異なる。

引数：

ctlp ： PDF ファイルのポインタ

lpSetEncryptOpt ： セキュリティオプション（詳細は説明参照）

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_EncryptOptionTD の定義を以下に示す。

```
typedef struct {
char            UserPWord[PL_PASSWORD_LEN + 1];     // ユーザパスワード
char            OwnerPWord[PL_PASSWORD_LEN + 1];    // マスタパスワード
PL_PermissionE  Permission;                         // 権限
int             Length;                             // 暗号化キーの長さ
PL_RevisionE    Revision;                           // 暗号化バージョン番号
} PL_EncryptOptionTD;
```

- PL_PASSWORD_LEN は最大パスワード長を示す。32 バイトである。

  #define PL_ PASSWORD_LEN 32

- UserPWord にはユーザパスワード（オープンパスワード）、OwnerPWord にはオーナパスワード（マスタパスワード）を設定する。この 2 種類には異なる文字列を指定する必要がある。また、使用可能な文字コードは 0x20～0x7E の範囲とする。
- Revision には暗号化に使用するプログラムのバージョンを設定する。PDF1.3 では指定にかかわらず 2 を使用する。PDF1.4、PDF1.5 では 2 または 3 が指定可能である。PDF1.6 以上では 2、3 または 4 が指定可能である。128 ビット暗号化を使用する場合、または Revision 3 で使用可能となる権限設定を必要とする場合、PDF のバージョンは 1.4 以上、Revision には 3 を指定する必要がある。AES256 ビットを利用する場合は 5 を指定する必要がある。

  ```
  typedef enum {
      PL_REVDEFAULT = 0, PL_REV2 = 2, PL_REV3, PL_REV4, PL_REV5
  } PL_RevisionE;
  ```

- Length は暗号化キーの長さを指定する。PDF1.3 では 40 ビット固定であるため、指定にかかわらず 40 ビットを使用する。PDF1.4 以降では Revision 指定が 2 の場合は、指定にかかわらず 40 ビットを使用する。Revision が 3 または 4 の場合、40～128 の間の 8 の倍数の値が指定可能である。この条件外の値が指定された場合、128 ビットを使用する(128 を推奨)。
- Permission は 32 ビットの整数で、文書をユーザパスワードで開いた場合に許可するアクセス権限を指定する。Revision の違いにより、この変数の意味も異なる。以下に、1(下位)~32(上位)のビットの意味を示す。
  - ✧ Revision 2 (PDF1.3、PDF1.4、PDF1.5、PDF1.6 で使用可能)の場合のビット定義

    ビット　位置の意味

    1–2　　予約：必ず 0 のこと

3　　文書の印刷可

4　　(テキスト注釈および対話フォームフィールド以外の)文書内容の変更可

5　　文書からテキストとグラフィックスのコピー可

6　　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可

7–32　予約：必ず 1 のこと

✧ Revision 3(PDF1.4、PDF1.5、PDF1.6 版で使用可能)の場合のビット定義

ビット　位置の意味

1–2　予約：必ず 0 のこと

3　低解像度の印刷を許可する

4　(テキスト注釈及び対話フォームフィールド以外の) 文書内容変更可

5　ビット 10 で制御される以外の操作による文書からテキストとグラフィックスのコピーと抽出可

6　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可

7-8　予約：必ず 1 のこと

9　ビット 6 がクリアされている場合も既存のフォーム（署名フィールドを含む）への入力を許可する

10　テキストとグラフィックスの抽出を許可する

11　文書のアセンブルを許可する

12+3　高解像度の印刷を許可する

13-32　予約、必ず 1 のこと

```
typedef enum {
        PL_ENABLE_ALL                = 0xFFFFFFFC,
        PL_ENABLE_NONE_R2            = 0xFFFFFFC0,
        PL_ENABLE_NONE_R3            = 0xFFFFF0C0,
        PL_ENABLE_PRINT              = 1 << 2,
        PL_ENABLE_CHGDOC             = 1 << 3,
        PL_ENABLE_COPY               = 1 << 4,
        PL_ENABLE_CHGANNOT           = 1 << 5
        //for Revision 3(used in PDF 1.4 or 1.5,not used in PDF1.3)
        PL_ENABLE_PRINTHIGHLEVEL     = (1 << 11) + PL_ENABLE_PRINT,
        PL_ENABLE_FILLFORM           = 1 << 8,
        PL_ENABLE_ACCESSIBILITY      = 1 << 9,
        PL_ENABLE_ASSEMBLEDOC        = 1 << 10
} PL_PermissionE;
```

- PDF/X、PDF/A：

  PDF/X、PDF/A では、パスワード設定は許可されない。

8.2.3. 圧縮設定関数

PDFAPI void pl_PutCompressOpt2(hPDF ctlp,PL_CompressOption2TD& lpSetCompressOpt)

機能：

イメージおよびコンテントストリームの圧縮方法を指定する。イメージについては、カラーイメージ、グレースケールイメージ、白黒イメージのタイプ別に設定する。

引数：

ctlp ： PDF ファイルのポインタ

lpSetCompressOpt ： 圧縮オプション（詳細は説明参照）

戻り値：

無し

説明：

PL_CompressOpt2TD の定義を以下に示す。

```
typedef struct {
    PL_CGImgCompressTD      cImgCompress;    // カラーイメージ用設定
    PL_CGImgCompressTD      gImgCompress;    // グレースケールイメージ用設定
    PL_MImgCompressModeE    mImgFMode;       // モノクロイメージ用設定
    PL_TxtCompressModeE     tlFMode;         // テキストとラインアート圧縮設定
} PL_CompressOption2TD;
```

- cImgCompress,gImgCompress では、カラー、およびグレースケールイメージの圧縮方法を設定する。以下を指定する。

```
typedef struct {
    PL_ImgCompressModeE    cgImgFMode;      // 圧縮方法
    int                    JpegQuality;     //  JPEG 画質(0～100)
} PL_CGImgCompressTD;

typedef enum {
    PL_IMGCM_NO = -1,
    PL_IMGCM_AUTO,
    PL_IMGCM_JPEG,
    PL_IMGCM_ZLIB,
    PL_IMGCM_AUTO2K,
    PL_IMGCM_JPEG2K
} PL_ImgCompressModeE;
```

デフォルト値は PL_IMGCM_AUTO である。

pl_PutImage 関数で BMP イメージ、およびイメージパススルーを使用してそのまま PDF に格納することができないイメージが出力された場合、PL_ IMGCM _JPEG 圧縮では JPEG

圧縮、PL_ IMGCM _ZLIB 圧縮では Flate 圧縮、PL_ IMGCM _JPEG2K では、JPEG2000(JPX)圧縮を行って、PDF に出力する。PL_IMGCOMPRESS_AUTO 圧縮では JPEG と ZLIB 双方の圧縮、PL_ IMGCM _JPEG2K では JPEG2000 と ZLIB 双方の圧縮を圧縮を試み、サイズが小さくなる側を採用する。ただし、JPEG 変換ができないイメージでは FLATE 圧縮を行って格納する。また、PDF1.4 以下では JPEG2000 は使用できないため、AUTO2K では AUTO、JPEG2K では JPEG が指定されたものとして動作する。

注：PL_ IMGCM _AUTO、同 AUTO2K は 2 種類の圧縮を試みるため、出力速度が遅くなる。

イメージパススルーで格納可能なイメージの場合、基本的には、この指定に関係なく、Flate 圧縮を行って、出力する。

JpegQuality はイメージの圧縮品質を指定する。BMP を JPEG、JPEG2000 圧縮する場合のみ参照される。値は 0 から、100 までが指定可能であり、値が大きくなるほど、画質が良い。デフォルトは 80 である。

- mImgFMode ではモノクロイメージの圧縮方法を指定する。以下から選択する。

```
typedef enum{
        PL_IMG CM _M_CCITT4=0,          // CCITTFax デコード Gr.4 フィルタ
        PL_IMG CM _M_CCITT3,            // CCITTFax デコード Gr.3 フィルタ
        PL_IMG CM _M_RUNLENGTH ,        // RunLength デコードフィルタ
        PL_IMG CM _M_ZLIB,              // Flate デコードフィルタ
        PL_IMG CM _M_OFF                //  フィルタ無し(非圧縮)
} PL_MImgCompressModeE;
```

- TextAndLineArt はテキストとグラフィックの圧縮方法を設定する。この変数の定義を以下に示す（サポートされる圧縮方法は、Flate 圧縮のみである）。

```
typedef enum {
    PL_TXTCM_NO = 0,   PL_TXTCM_ZLIB
} PL_TxtCompressModeE;
```

8.2.4. ASCII 形式出力設定関数

PDFAPI void pl_PutASCIIFilter(hPDF ctlp, HBOOL bASCIIFile)

機能：

ASCII85 エンコーディングで PDF ファイルを出力する場合に設定する。

引数：

ctlp ： PDF ファイルのポインタ

bASCIIFile：HFALSE　ASCII85 エンコーディングを使用しない。(デフォルト)

HTRULE　ASCII85 エンコーディングを使用する。

戻り値：

無し

8.2.5. 画像比較設定関数

PDFAPI void pl_SetImgCompOpt(hPDF ctlp,PL_ImageCompModeE imgCompMode)

機能：

ファイルパスによる画像出力時は、同一パスである画像が存在すれば、ダウンサンプリング指定により、画像が加工される場合を除き、その画像オブジェクトを共用するように出力する。ストリーム出力による画像指定時、パスによる判定を行うことができないため、画像内容を比較して、同一画像であるか否かを確認する必要がある。本関数は、この比較を行うか否かを指定する。通常、完全比較を指定のこと。

引数：

ctlp ： PDF ファイルのポインタ

imgCompMode：比較方法の指定

PL_ImageCompModeE の定義を以下に示す。

```
typedef enum {
        PL_ICM_NONE = 0,          // 比較しない
        PL_ICM_APPROXIMATE,  // 簡単な比較
        PL_ICM_PERFECT            // 完全比較
} PL_ImageCompModeE;
```

戻り値：

無し

8.2.6. オブジェクト圧縮設定関数

PDFAPI void pl_SetObjOpt(hPDF ctlp, HBOOL bObjStream)

機能：

オブジェクトレベルの圧縮(PDF1.5 以降)を行うか否かを指定する

引数：

ctlp：PDF ファイルのポインタ

bASCIIFile：HFALSE　　オブジェクトレベルの圧縮を行わない（デフォルト）

HTRULE　　オブジェクトレベルの圧縮を行う

戻り値：

無し

8.2.7. 文書情報設定関数

PDFAPI PL_ERROR pl_SetDocInfo(hPDF ctlp, PL_InfoTD * infop)

機能：

PDF ファイルの文書属性辞書の内容を設定する。Author、Creator、Title、Subtype、Keywords が設定できる。

引数：

ctlp ： PDF ファイルのポインタ

infop： 文書情報を格納する構造体へのポインタ。PDF ファイルに設定する文書情報をこの構造体に設定する。構造体については説明参照。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

この関数を使う場合、必ず pl_OpenPdf()関数を使う前に使わなければならない。

PL_InfoTD 構造体の定義を以下に示す。

```
typedef struct {
    UCHAR_t     *Author;        // 著者
    UCHAR_t     *Creator        // 生成者(Application 名等)
    UCHAR_t     *Title;         // タイトル
    UCHAR_t     *Subject;       // 副タイトル
    UCHAR_t     *Keywords;      // キーワード
} PL_InfoTD;
```

PDF/X では、必ず Title を設定する必要がある

8.2.8. ベース URI 設定関数

PDFAPI PL_ERROR pl_SetBaseUri(hPDF ctlp, const char* BaseUri)

機能：

PDF の文書カタログから参照される URI 辞書の Base エントリを設定する。この関数で BaseUri を設定することにより、リンク先を指定する相対パスに対して、ビューアでは絶対パスを作成する。

引数：

ctlp ： PDF ファイルのポインタ

BaseUri ： ベース URI を設定する

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.2.9. ページモード設定関数

PDFAPI void pl_PutPageModeOpt(hPDF ctlp, PL_PageModeE PageMode)

機能：

PDF の文書カタログの PageMode エントリを設定する。このエントリは PDF ファイルを開いたときのパネルの表示方法の指定となる。

引数：

ctlp ： PDF ファイルのポインタ

PageMode ： PDF ファイルを開いたときの表示モードを設定する。説明参照。

戻り値：

なし

説明：

PageMode の定義を以下に示す。

```
typedef enum {
    PL_PM_OUTLINEFREE = 0,PL_PM_USENONE,PL_PM_USEOUTLINES,
    PL_PM_USETHUMBS,PL_PM_FULLSCREEN,PL_PM_USEOC
} PL_PageModeE;
```

- PL_ PM_OUTLINEFREE: Outline がある場合、UseOutlines を設定する。Outline がない場合、文書情報辞書の PageMode エントリを作成しない。Acrobat の場合、PageMode エントリがない文書は Acrobat の環境設定にしたがって表示される(デフォルト値)。
- PL_ PM_USENONE: UseNone を設定する。PDF のオープン時、Outline、Thumbnail パネルが表示されない状態になる。
- PL_ PM_USEOUTLINES: UseOutlines を設定する。PDF のオープン時、Outline パネルが表示された状態となる。
- PL_ PM_USETHUMBS: UseThumbs を設定する。PDF のオープン時、Thumbnail パネルが表示された状態となる(本ライブラリでは、Thumbnail の作成をサポートしない)。
- PL_FULLSCREEN: FullScreen を設定する。全画面表示になり、メニューバー、ウィンドウコントロールバーなどが表示されない状態になる。
- PL_ PM_USEOC: PDF1.5 以降であれば UseOC を設定する。PDF のオープン時、オプショナルコンテント(レイヤー)パネルが表示された状態となる(本ライブラリでは、オプショナルコンテントの作成をサポートしない)。PDF1.4 以下の場合、PL_ PM_OUTLINEFREE と同様となる。

8.2.10. ページレイアウト設定関数

PDFAPI void pl_PutPageLayoutOpt(hPDF ctlp, PL_PageLayoutE PageLayout)

機能：

PDF の文書カタログの PageLayout エントリを設定する。このエントリは文書を開いたときに使われるページレイアウトを設定するものである。

引数：

ctlp ： PDF ファイルのポインタ

PageLayout ： PDF ファイルが開いた時のページレイアウトを設定する。説明参照。

戻り値：

なし

説明：

PageLayout の定義を以下に示す。

```
typedef enum {
```

PL_PL_NONE = 0, PL_PL_SINGLEPAGE, PL_PL_ONECOLUMN,
PL_PL_TWOCOLUMNLEFT, PL_PL_TWOCOLUMNRIGHT,PL_
PL_TWOPAGELEFT, PL_PL_TWOPAGERIGHT

} PL_PageLayoutE;

以下に、処理を示す。

- PL_PL_NONE: PageLayout エントリを出力しない(デフォルト値)。
- PL_ PL_SINGLEPAGE: SinglePage を設定する。1 度に 1 ページを表示する。
- PL_ PL_ONECOLUMN: OneColume を設定する。1 ページに 1 列で表示する。
- PL_ PL_TWOCOLUMNLEFT: TwoColumnLeft を設定する。奇数ページを左側にして、2 列で表示する。
- PL_ PL_TWOCOLUMNRIGHT: TwoColumnRight を設定する。奇数ページを右側にして 2 列で表示する。
- PL_ PL_TWOPAGELEFT: TwoPageLeft を設定する。偶数ページを左側にして 2 ページを表示する(PDF1.5 以降)。
- PL_ PL_TWOPAGERIGHT:TwoPageRight を設定する。偶数ページを右側にして 2 ページを表示する (PDF1.5 以降)

8.2.11. ビューアプレファレンス設定関数

PDFAPI PL_ERROR pl_PutViewPreferences(hPDF ctlp,PL_ViewPrefTD* lpViewPref)

機能：

文書カタログから参照されるビューアプレファレンス辞書をする。これは画面上の文書の表示方法を制御するものである。

引数：

ctlp ： PDF ファイルのポインタ

lpViewPref ： PDF ファイルを開く時の画面上の表示方法を設定する

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

lpViewPref の定義を以下に示す:

typedef struct {

PL_WindowModeE winMode; // ビューアのパラメータの設定。

PL_ScreenModeE NonFullScrPgMode;

// 全画面表示モードを抜けたときの表示

// 詳細は説明参照

PL_DirectionE Direction; // テキストを読み進める際のページをめくる順序

// 詳細は説明参照

PL_VPBoxE ViewArea; // プリプレス用表示領域 (PDF1.4)

```
    PL_VPBoxE    ViewClip;          // プリプレス用表示クリップ領域(PDF1.4)
    PL_VPBoxE    PrintArea;         // プリプレス用印刷領域 (PDF1.4)
    PL_VPBoxE    PrintClip;         // プリプレス用印刷クリップ領域(PDF1.4)
} PL_ViewPrefTD;
```

- PL_WindowModeE：以下の値を設定する。

```
typedef enum {
        PL_WM_NONE                = 0,
        PL_WM_HIDETOOLBAR         = 1 << 0,        // ビューアアプリケーションのツール
                                                   // バーを隠す(デフォルト値は false)。
        PL_WM_HIDEMENUBAR         = 1 << 1,        // ビューアアプリケーションのメニュ
                                                   // ーバーを隠す(デフォルト値は false)。
        PL_WM_HIDEWINDOWUI        = 1 << 2,        // ウィンドウのユーザインタフェース
                                                   // 要素(スクロールバー、ナビゲーション用
                                                   // コントロールなど)を隠す（デフォルト値は false)。
        PL_WM_FITWINDOW           = 1 << 3,        // ウィンドウを最初に表示されるペー
                                                   // ジのサイズに適合するようにサイズ
                                                   //変更する（デフォルト値は false)。
        PL_WM_CENTERWINDOW        = 1 << 4,        // ウィンドウを画面の中央に配置する
                                                   //（デフォルト値は false）
        PL_WM_DISPLAYDOCTITLE = 1 << 5             // ウィンドウタイトルバーにファイル
                                                   // 名ではなく文書情報辞書の Title エン
                                                   // トリの内容を表示する（デフォルト値は false）
} PL_WindowModeE;
```

- NonFulleScrPgMode：pl_PutPageMode で、PL_FULLSCREEN を指定した文書で、フルスクリーン表示を抜けたときの文書表示方法となる。デフォルトは PL_FS_USENONE である。

  PL_ScreenMode の定義を以下に示す。

```
typedef enum{
    PL_FS_USENONE=0,PL_FS_USEOUTLINES,PL_FS_USETHUMBS,
    PL_FS_USEOC
}PL_ScreenModeE;
```

  それぞれの意味を以下に示す。

  - PL_FS_USENONE: UseNone を設定する。Outline パネル、Thumbnail パネルいずれも表示されない。
  - PL_FS_USEOUTLINES: UseOutlines を設定する。Outline パネルが表示される。
  - PL_FS_USETHUMBS: UseThumbs を設定する。Thumbnail パネルが表示される。
  - PL_FS_USEOC:UseOc を設定する。Optional Contents パネルが表示される(PDF1.5 以降)。

- Direction：テキストの読込み順序を記述する。PL_Direction の定義を以下に示す。

  typedef enum{

  PL_D_L2R=0,PL_D_R2L

  }PL_DirectionE;

  それぞれの意味を以下に示す。

  - PL_D_L2R: 左から右へ
  - PL_D_R2L: 右から左へ (中国語、日本語、韓国語の縦書きを含む)
- ViewArea：画面表示時のページ上の表示領域を、ページオブジェクトに定義される領域の名前で指定する。PDF1.4 以降の機能。PL_VPBoxE の説明参照
- ViewClip：画面表示のページコンテンツがクリップされる領域を、ページオブジェクトに定義される領域の名前で指定する。PDF1.4 以降の機能。PL_VPBoxE の説明参照
- PrintArea：印刷時のページ上の表示領域を、ページオブジェクトに定義される領域の名前で指定する。PDF1.4 以降の機能。PL_VPBoxE の説明参照
- PrintClip：印刷時のページコンテンツがクリップされる領域を、ページオブジェクトに定義される領域の名前で指定する。PDF1.4 以降の機能。PL_VPBoxE の説明参照

PL_VPBoxE ：PDF のページオブジェクトに MediaBox,CropBox,BleedBox,TrimBox、ArtBox の 5 種類の領域が定義される。上記の 4 領域はこれらの中のいずれの領域を、それぞれの対象領域とするかを選択するものである。PL_VPBoxE の定義を以下に示す。

```
typedef enum {
    PL_VP_CROPBOX = 0, PL_VP_MEDIABOX, PL_VP_BLEEDBOX,
    PL_VP_TRIMBOX, PL_VP_ARTBOX
} PL_VPBoxE;
```

- PDF/X：

  ViewArea は PL_MEDIABOX 或は PL_BLEEDBOX を使用しなければならない。 ViewClip、 PrintArea、PrintClip も同様である。

8.2.12. オープンアクション設定関数

PDFAPI void pl_PutOpenAction(hPDF ctlp,PL_DestTD* lpOpenAction)

機能：

文書カタログの Open Action エントリを設定する。PDF ファイルを開いた時に実行されるアクションを指定するエントリであるが、本関数は、表示するページを指定する機能だけをサポートする。

引数：

ctlp：PDF ファイルのポインタ

lpOpenAction：PDF ファイルを開いた時に表示される位置を設定する。PL_DestTD については共通データ形式の説明参照。

戻り値：

なし

説明：

設定したページが PDF ファイルに存在しない場合、その設定は無効となり、無視される。

8.2.13. オープンアクション設定関数 2

PDFAPI PL_ERROR pl_PutOpenActionByHandle(hPDF ctlp,PL_ActionHandle hAction)

機能：

文書カタログの Open Action エントリを設定する。PDF ファイルを開いた時に実行されるアクションを指定する。後述のアクションの作成関数により作成・取得したアクションのハンドルを指定する。

引数：

ctlp：PDF ファイルのポインタ

hAAction：PDF ファイルを開いた時に実行するアクションを設定する。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.2.14. 事前定義された名前によるオープンアクション設定関数

PDFAPI void pl_PutOpenActionByName(hPDF ctlp,const unsigned char* lpDestName)

機能：

文書カタログの Open Action エントリを設定する。PDF ファイルが開いた時に実行されるアクションを指定する。本関数では、名前指定によるアクションの指定だけをサポートする。

引数：

ctlp：PDF ファイルのポインタ

lpDestName：PDF ファイルを開いた時に表示される位置を設定する。該当名称は定義した位置である。

戻り値：

なし

説明：

ここで設定される名称については、本ライブラリでは特にチェックを行わない。指定された名称の動作については、PDF を開いたアプリケーションに依存する。

8.2.15. ページラベル設定関数

PDFAPI PL_ERROR pl_PutPageLabelOpt(hPDF ctlp,int PgNo,PL_PageLabelE LabelType,
const UCHAR_t* lpPrefix,int StartNum)

機能：

文書カタログの PageLabels エントリに PDF のページラベルを設定する。

引数：

ctlp：PDF ファイルのポインタ

PgNo ：ページ番号（このラベルを使用する先頭ページ。0 オリジン）

LabelType ：数値部分の番号付けスタイルを指定する

typedef enum{

PL_PgLabel_NO=0,PL_PgLabel_D,PL_PgLabel_R,PL_PgLabel_r,

PL_PgLabel_A,PL_PgLabel_a

} PL_PageLabelE;

定義を以下に示す。

PL_PgLabel_NO： 数値部分を出力しない

PL_PgLabel_D： 10 進アラビア数字

PL_PgLabel_R： 大文字のローマ数字

PL_PgLabel_r： 小文字のローマ数字

PL_PgLabel_A： アルファベットの大文字

PL_PgLabel_a： アルファベットの小文字

lpPrefix ：この範囲のページラベルに用いるラベルプリフィックス

StartNum ：この範囲の先頭ページラベルに用いる数値部分の値（1 以上の値）

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.2.16. 名前付き宛先定義関数

PDFAPI PL_ERROR pl_SetNames(hPDF ctlp, const unsigned char* lpNames,PL_DestTD* lpDest)

機能：

リンク注釈、アウトラインで使用される GoTo,GoToR アクションの宛先として使用できる名前付き宛先の定義機能を定義する。

引数：

ctlp：PDF ファイルのポインタ

lpNames：宛先の名前を指定。文字列の規約は PDF の名前ツリーの定義による

lpDest：共通データ形式の説明参照

宛先種別：8 種類のタイプ(DEST_XYZ など)から 1 つを指定する

宛先パラメータ：宛先種別ごとに規定される値を指定する(float a,b,c,d)

ページ番号：整数（1 オリジンとし、0 でカレントページとなる。PDF の仕様の 0 オリジンとは異なるため注意）

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.2.17. PDF/X エラー処理設定設定関数

PDFAPI PL_ERROR pl_SetPdfXOpt (hPDF ctlp,PL_InvalidPdfXActionE invalidPdfXAct)

機能：

PDF/X に適合しない機能が指定された場合の動作を設定する

引数：

ctlp ：PDF ファイルポインタ

invalidPdfXAct：PDF/X に適合しない機能が指定された場合の動作を設定する。定義を以下に示す。

```
typedef enum {
        PL_InvalidPdfX_RetError    = 0,              // エラーを戻す(デフォルト)
        PL_InvalidPdfX_Continue = 1 << 1,            // エラーを無視して継続する
        PL_InvalidImpPdfX_Continue = 1 << 2          // インポートが指定された
                                                     // PDF ファイルについてのエラーのみ無視して継続する
} PL_InvalidPdfXActionE;
```

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す。

説明：

この関数は pl_Intital()の後、pl_SetPdfVersion の前に設定すること。

8.2.18. 言語設定関数

PDFAPI PL_ERROR pl_SetDocumentLang(hPDF ctlp, const char* lang);

機能：

タグ付 PDF の言語設定を指定する

引数：

ctlp ：PDF ファイルポインタ

lang：言語設定

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す。

8.2.19. タグ付 PDF エラー処理設定関数

PDFAPI PL_ERROR pl_SetTaggedPdfOpt(hPDF ctlp,PL_InvalidTaggedPDFActionE invalidTaggedPDFAct);

機能：

タグ付 PDF のタグの不整合を検出した場合の動作を指定する

引数：

ctlp ：PDF ファイルポインタ

invalidTaggedPDFAct：タグの不整合を検出した場合の動作を指定する。定義を以下に示す。

```
typedef enum {
        PL_InvalidTaggedPDF_RetError = 0,            // エラーを戻す(デフォルト)
        PL_InvalidTaggedPDF_Continue = 1 << 1,       // エラーを無視して継続
        PL_InvalidTaggedPDF_ImportContinue = 1 << 2          // インポートが指定された
```

// PDF ファイルについてのエラーのみ無視して継続する

} PL_InvalidTaggedPDFActionE;

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す。

説明：

この関数は pl_Intital()の後、pl_SetPdfVersion の前に設定すること。

8.2.20. タグ付 PDF 設定関数

PDFAPI void pl_SetTaggedPdfMode(hPDF ctlp,HBOOL bTagged=HFALSE);

機能：

タグ付 PDF の出力を行うか否かを指定する。

引数:

bTagged：HTRUE　　タグ付 PDF を出力する

HFALSE　　通常の PDF を出力する(デフォルト)

戻り値：

なし

説明：

この関数は pl_Intital()の後、pl_SetPdfVersion の前に設定すること。

8.2.21. 埋め込みフォント設定関数

PDFAPI PL_ERROR pl_PutEmbOpt(hPDF ctlp,PL_EmbedOptionTD* lpSetEmbOpt)

機能：

フォント埋め込みのオプションを設定する。埋め込み指定が可能なフォントは Type1 フォント(pfb ファイルが必要)、TrueType フォント、および OpenType フォントである。また、フォントベンダが埋め込みを禁止しているフォントは対象外となる。

引数：

ctlp ： PDF ファイルのポインタ

lpSetEmbOpt ： 埋め込みオプション（詳細は説明参照）

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_EmbedOptionTD 構造体の定義を以下に示す。

```
typedef struct {
    PL_EmbedFontInfoTD   EmbedFontInfo;   // 埋め込むフォントの指定
    HBOOL        CantEmbedAction;         // 埋め込みできないフォントの処理指定
    HBOOL        UnusualAction;           // 標準 CMap で表現できない文字の処理指定
    HBOOL        SymbolFontAction;        // Symbolic フォントの処理
```

```
    HBOOL          Base14Embeded;          // Base14 フォントの埋込み
    HBOOL          bFullEmbedBasePercent;  // フルセット埋め込み指定
    unsigned char  fullEmbedBasePercent;   // フルセット埋め込みのパーセント指定(0～100)
} PL_EmbedOptionTD;
```

- EmbedFontInfo ： フォントの埋め込みに関する処理を指定する。

EmbedFontInfoTD 構造体の定義を以下に示す。

```
typedef struct {
    PL_EmbedTypeE    EmbedType;     // 埋込みタイプ指定
    PL_EmbedTblTD    EmbedTbl;      // 埋込フォントのテーブル
} PL_EmbedFontInfoTD;
```

EmbedType の定義を以下に示す。

```
typedef enum{
    PL_ET_PARTEMBED=0,              // EmbedTbl に指定されたフォントだけを埋込む
    PL_ET_ALLEMBED                  // EmbedTbl に指定されたフォントだけを埋込む
}PL_EmbedTypeE;
```

PL_ ET_ALLEMBED を指定する場合、EmbedTbl の指定は不要である。

EmbedTbl ：埋め込みを行うフォントを指定する。EmbedType に PL_ ET_PARTEMBED が設定されている場合、このテーブルが参照される。

PL_EmbedTblTD の定義を以下に示す

```
typedef struct {
    int                 CurSize;    // 埋め込みフォントのテーブルのサイズ
    PL_EmbedFontTD      *lpEmbFont; // 埋め込みフォントテーブルへのポインタ
} PL_EmbedTblTD;
```

PL_EmbedFontTD の定義を以下に示す。

```
typedef struct {
    UCHAR_t  UFName[PL_NAMELEN]; // フォント名
    int      Weight;             // ウェイト(未サポート、設定不要)
    HBOOL    bItalic ;           // イタリック(未サポート、設定不要)
                                 //  (HFALSE : 非イタリック、HTRUE : イタリック)
} PL_EmbedFontTD;
```

説明：埋め込むを行うフォントの名称を設定することで、全ての同じファミリー名を持つフォントが埋め込まれる（Weight と bItalic による区別は現在サポートしていない。従って、Bold のみを埋め込み、Regular は埋め込まない、というような指定はできない）。

- CantEmbedAction ： 埋め込み指定されたフォントが、埋め込みを未サポートのフォント形式であった場合、あるいはフォントベンダが埋め込みを禁止しているフォントであった場合の動作を指定する。定義を以下に示す。

  #define PL_UNEMBEDABLE_CONTINUE　　HFALSE
  　　// 埋め込みを行わず、処理を続行する(デフォルト)
  #define PL_UNEMBEDABLE_RTERROR　　HTRUE
  　　// エラー(処理を中止する)

- UnusualAction ：出力文字中に、通常の CMap またはエンコーディングで表現できない文字、またはフォントファイルに存在しない文字が検出された場合の動作を指定する。定義を以下に示す。

  #define PL_EMBEDTRUE　　HTRUE　　// 埋め込む(デフォルト)
  #define PL_EMBEDFALSE　　HFALSE　　// 埋め込みしない

  PL_EMBEDTRUE では、それらの文字のグリフの埋め込みを行う。PL_EMBEDFALSE が指定された場合、グリフ番号のみを PDF に出力する。

  注：グリフ番号のみを出力した場合、フォントが存在する環境でも PDF の文字が正しく表示されない現象が発生することがある。

- SymbolFontAction： 埋め込みフォントの指定にかかわらず、TrueType、OpenType のシンボリックフォント、および Type1 の FontSpecific エンコーディングフォントの埋め込みを行うか否かを指定する。

  値の定義は UnusualAction と同じである。

- Base14Embeded：本ライブラリでは、標準 14 フォントは、通常、EmbedFontInfo の設定によらず埋め込み処理を行わない。この設定を true とした場合に、標準 14 フォントを他のフォント動揺の埋め込み指定の対象となる。

8.2.22. ミッシンググリフ処理設定関数

PDFAPI void pl_PutMissGlyphOpt(hPDF ctlp, HBOOL MissTxtOpt)

機能：

ミッシンググリフ(指定されたフォントにグリフが定義されていない文字)を検出した場合、この関数のオプションの設定で、その文字を置換して続けて実行するかエラー戻すかを指定することができる。

引数：

ctlp ： PDF ファイルのポインタ

MissTxtOpt：ミッシンググリフに対してどのように処理するかを指定する。説明参照。

戻り値：

なし

説明：

MissTxtOpt：フォントファイルのグリフが存在しない文字があった場合の処理を指定する。

#define PL_MISS_REPLACE　　HFALSE

// デフォルト文字で置き換えて処理を続行する(デフォルト)

#define PL_MISS_RTERROR　　HTRUE

// 処理を打ち切り、エラーを戻す

注:デフォルト文字は通常、フォント内で定義される。TrueType、OpenType CID フォントでは OS/2 テーブルの DefaultChar で指定される文字を使用する。Type1 および OpentypeNONCID フォントでは空白文字を使用する。これらの定義・文字が存在しないフォントの場合、グリフ番号 0 の文字を使用する。

8.2.23.　Identity 設定関数

PDFAPI void pl_PutEncodeOpt(hPDF ctlp, HBOOL UseIdenFlag)

機能：

埋め込みを行なわない場合、普通の CMap を使って正確に表示できない文字を暗黙的にグリフ番号出力とするか否かを指定する。

引数：

ctlp ： PDF ファイルのポインタ

UseIdenFlag ： Identity-H(V)のオプションを使う（詳細は説明参照）

戻り値：

なし

説明：

UseIdenFlag： Identity-H(V)のマークを使用するかどうかを設定する。

定義を以下に示す。

#define PL_USEIDENTITY　　HTRUE

// 0x5c(J,K)と 0x7e(J)に対して Idnetity-H(V)を使う(デフォルト値)

#define PL_NOTUSEIDENTITY　HFALSE

// 0x5c(J,K)と 0x7e(J)に対して Idnetity-H(V)を使用しない。

注:日本語の u+005c に対して¥を割り当てているフォントと＼(Backslash)を割り当てているフォントが存在し、通常の文字コード出力を行った場合、Windows 上で所定の文字が表示されない現象を検出したために、設けたオプションである。

ここで PL_USEIDENTITY を設定し、前述の EmbedOptionTD の UnsusualAction で PL_EMBEDTRUE を指定することで、これらの文字の埋め込みが行われる。韓国語の u+005c についても同様の問題がある。

8.2.24.　イメージダウンサンプリング設定関数

PDFAPI void pl_PutDownSampling(hPDF ctlp, const PL_ImgDownSampleTD& imgDownSample)

機能：

イメージのダウンサンプリングに関するパラメータを設定する。

引数：

ctlp ：PDF ファイルのポインタ

imgDownSample：ダウンサンプリング用パラメータ

戻り値：

無し

説明：

imgDownSample にてカラーイメージ、グレースケールイメージ、モノクロイメージそれぞれのダウンサンプリングの可否、設定を行う。PL_ ImgDownSampleTD の定義を以下に示す。

```
typedef struct
{
    PL_ DownSampleTD cImgDownSample;      // カラーイメージダウンサンプリングモード
    PL_ DownSampleTD gImgDownSample;      // グレースケールイメージダウンサンプリングモード
    PL_ DownSampleTD mImgDownSample;      // モノクロイメージダウンサンプリングモード
} PL_ ImgDownSampleTD;
```

ダウンサンプリング手法は、無し、アベレージダウンサンプリング(バイリニア)、バイキュービックダウンサンプリング、サブサンプリング(ニアレストネイバー)のいずれかから選択する。

```
typedef enum{
    PL_DS_NONE=0,                 // ダウンサンプリングしない
    PL_DS_AVERAGE,                // アベレージダウンサンプリング
    PL_DS_BICUBIC,                // バイキュービックダウンサンプリング
    PL_DS_SUBSAMPLING             // サブサンプリング
} PL_DSModeE;
```

また、ダウンサンプリング対象とする最小 ppi 値、および、その場合のダウンサンプリング解像度を下記で設定する。

```
typedef struct
{
    PL_DSModeE    dsMode;         // ダウンサンプリングモード
    int           aboveDPIValue;  // ダウンサンプリング対象とする ppi 値下限
    int           toDPIValue;     // ダウンサンプリング ppi 値
} PL_DownSampleTD;
```

説明：

ダウンサンプリングの指定により、イメージデータの加工が行わる。この過程で、イメージのカラースペースが変更される場合がある。

8.2.25. 透過イメージの処理方法の設定関数

PDFAPI void pl_SetTransImgProcMode(hPDF ctlp,PL_ImgProcModeE mode)

機能：

透明属性が指定されているイメージの処理方法を設定する

引数：

ctlp ： PDF ファイルのポインタ

mode：イメージの透明処理方法。説明参照.

戻り値：

なし

説明：

指定された PDF バージョンで透過属性を使用したイメージに対応できない場合の処理を指定する。PDF1.3 の場合、αチャネルを持つ TIFF、PNG イメージが該当する。PDF1.4 以降では本関数の設定によらずαチャネルを出力する。

- mode：

PL_ImgProcMode の定義を以下に示す。

```
typedef enum{
    PL_Transparency_NotProc=0, PL_TransparancyNeedProc
} PL_ImgProcModeE;
```

mode：

PL_Transparency_NotProc：PDF1.3 形式でのファイル出力時、αチャネルを使用した PNG と TIFF を未サポートイメージとする(デフォルト)。後述する pl_CheckImage または pl_CheckImage_Stream で該当するイメージに対して未サポートのステータスが戻る。

PL_Transparency_NeedProc：αチャネルを使う PNG と TIFF をサポートイメージとする。PDF1.3 ではαチャネルがサポートされないため、不透過イメージとして出力される。

8.2.26. イメージ (PNG、GIF、TIFF、JPG) のパススルー抑止設定関数

PDFAPI void pl_SetNoPassOpt(hPDF ctlp, PL_GSTypeE gTypes)

機能：

イメージ(PNG、GIF、TIFF、JPG)のタイプ別にパススルー禁止を設定する。

引数：

ctlp： PDF ファイルポインタ

gTypes ：パススルー禁止に設定されるイメージの種類。詳しくは説明参照。

戻り値：

無し

説明：

gTypes の定義を以下に示す。

```
typedef enum {
        PL_GST_NONE   = 0,
        PL_GST_GIF    = 1 << 0,        // GIF イメージ
        PL_GST_TIFF   = 1 << 1,        // TIFF イメージ
        PL_GST_PNG    = 1 << 2,        // PNG イメージ
        PL_GST_JPG    = 1 << 3,        // JPG イメージ
```

```
    PL_GST_JPG2K   = 1 << 4,          // JPEG2000 イメージ
    PL_GST_JBIG2   = 1 << 5           // JBIG2 イメージ
} PL_GSTypeE;
```

8.2.27. αチャネル付き 1 ビット GIF イメージの変換設定関数

PDFAPI void pl_SetNoBitConvOpt(hPDF ctlp,HBOOL isNoConv)

機能：

αチャネル付き 1Bit の GIF イメージの 8bits への変換を禁止するか否かを設定する。αチャネル付き 1Bit の GIFf イメージは、そのままでは Acrobat 5.0 で開く場合にエラーが発生する。この問題を回避するために 8Bit に変換している。これを抑止する必要がある場合、本関数で指定する。

引数：

ctlp ： PDF ファイルポインタ

isNoConv ： αチャネル付き 1Bit の GIF イメージの 8Bit への変換の禁止を設定する場合 HTRUE を指定する。

戻り値：

無し

8.2.28. カラープロファイル付きイメージの処理モード設定関数

PDFAPI void pl_SetImageColorProfileOpt (hPDF ctlp,PL_CPActionE colorProfileAct,
PL_CPHandle hColorProfile,PL_InvalidCPActionE invalidColorProfileAct)

機能：

カラープロファイル付きイメージの処理モードを設定する。

引数：

ctlp：PDF ファイルのポインタ

colorProfileAct：カラープロファイル付きイメージの処理モード

hColorProfile：プロファイル差し替え時のカラープロファイルハンドル

invalidColorProfileAct：プロファイル差し替え時、不整合となった場合の処理モード

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

カラープロファイル付きのイメージデータが使用された場合の処理を選択する。PL_CPActionE の定義を以下に示す。

```
typedef enum{
    PL_CP_AUTO_NONE=0,
        /* カラープロファイル無しのイメージであれば、そのまま出力する。カラープロファイル
        付きであれば、そのカラープロファイルも出力する。*/
    PL_CP_AUTO_SETTED,
```

/* カラープロファイル無しのイメージであれば、hColorProfile で指定されるプロファイルを付加して出力する。カラープロファイル付きのイメージであれば、そのカラープロファイルを合わせて出力する。*/

PL_CP_NONE,

/*カラープロファイルを削除して出力する*/

PL_CP_SETTED

/* カラープロファイル付きのイメージ、カラープロファイル無しのイメージともに、hColorProfile で指定されるプロファイルを付加して出力する*/

} PL_CPActionE;

hColorProfile には、PL_CP_AUTO_SETTED、および PL_CP_SETTED 時に使用するカラープロファイルのハンドルを指定する。pl_LoadColorProfile()で取得する。

invalidColorProfileAct には、PL_CP_AUTO_SETTED、および PL_CP_SETTED 時に使用するよう指定されたカラープロファイルと、イメージが不整合であった場合の動作を指定する。定義を以下に示す。

```
typedef enum{
    PL_InvalidCP_Continue=0,        // エラーを無視し、継続する(Default value)
                                    // 指定のカラープロファイルは使用しない
    PL_InvalidCP_RetError           // 処理を中止しエラー戻す
}PL_InvalidCPActionE;
```

8.2.29. ヘッダコメント設定関数

PDFAPI void pl_SetHeaderComment(hPDF ctlp,const char* comment,int commentLen)

機能：

ヘッダコメントを設定する

引数：

ctlp ：PDF ファイルポインタ

comment：ヘッダコメント文字列。

commentLen：ヘッダコメント文字列長。

戻り値：

なし。

8.2.30. ヘッダコメント設定関数（ストリーム）

PDFAPI PL_ERROR pl_SetHeaderComment(hPDF ctlp,std::istream& commentData)

機能：

ヘッダコメントを設定する

引数：

ctlp ：PDF ファイルポインタ

commentData：ヘッダコメントの文字列ストリーム。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す。

8.2.31. XMP データ設定関数

PDFAPI void pl_SetXMPData(hPDF ctlp,const char* xmpInfo,int xmpInfoLen)

機能：

XMP データを設定する

引数：

ctlp ：PDF ファイルポインタ

xmpInfo：XMP データ。

xmpInfoLen：XMP データ長。

戻り値：

なし。

8.2.32. XMP データ設定関数（ストリーム）

PDFAPI PL_ERROR pl_SetXMPData(hPDF ctlp,std::istream& xmpInfo)

機能：

XMP データを設定する

引数：

ctlp ：PDF ファイルポインタ

xmpInfo：XMP データストリーム。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す。

8.2.33. PDF/A エラー処理設定設定関数

PDFAPI PL_ERROR pl_SetPdfAOpt (hPDF ctlp,PL_InvalidPdfAActionE invalidPdfAAct)

機能：

PDF/A に適合しない機能が指定された場合の動作を設定する

引数：

ctlp ：PDF ファイルポインタ

invalidPdfAAct：PDF/A に適合しない機能が指定された場合の動作を設定する。定義を以下に示す。

```
typedef enum {
        PL_InvalidPdfA_RetError    = 0,              // エラーを戻す(デフォルト)
        PL_InvalidPdfAX_Continue = 1 << 1,           // エラーを無視して継続する
        PL_InvalidImpPdfA_Continue = 1 << 2          // インポートが指定された
                                                     // PDF ファイルについてのエラーのみ無視して継続する
```

} PL_InvalidPdfAActionE;

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す。

説明：

この関数は pl_Intital()の後、pl_SetPdfVersion の前に設定すること。

8.2.34. PDF/A 出力時のカラースペース自動変換関数

PDFAPI PL_ERROR pl_SetPdfACSConvOpt(hPDF ctlp,HBOOL bConvCS)

機能：

PDF/A 出力時にカラースペースを自動変換するか否かを指定する

引数：

ctlp ：PDF ファイルポインタ

bConvCS：自動変換するか否かを指定する。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す。

8.2.35. PDF/X 出力時のカラースペース自動変換関数

PDFAPI PL_ERROR pl_SetPdfXCSConvOpt(hPDF ctlp,HBOOL bConvCS)

機能：

PDF/X 出力時にカラースペースを自動変換するか否かを指定する

引数：

ctlp ：PDF ファイルポインタ

bConvCS：自動変換するか否かを指定する。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す。

8.3. ページ制御関数

8.3.1. ページツリーノード作成関数

PDFAPI PL_ERROR pl_CreatePages(hPDF ctlp)

機能：

現在のページツリーノード内に新しいページツリーノードを作成する。以降のページは作成されたページツリーノードの下に配置される。この関数は必ずページを閉じてから使用すること。

引数：

ctlp ： PDF ファイルのポインタ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.3.2. ページツリーノードクローズ関数

PDFAPI PL_ERROR pl_ClosePages(hPDF ctlp)

機能：

現在のページツリーノードを閉じる。pl_CreatePages 関数を呼び出した場合、この関数を呼び出す。

引数：

ctlp ： PDF ファイルのポインタ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.3.3. ページオブジェクト作成関数

PDFAPI PL_ERROR pl_CreatePage(hPDF ctlp)

機能：

新しいページオブジェクトを作成する。カレントページツリーノードの下にページを作成する。必ず、前のページを閉じた後で呼び出すこと。

引数：

ctlp ： PDF ファイルのポインタ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.3.4. ページオブジェクトクローズ関数

PDFAPI PL_ERROR pl_ClosePage(hPDF ctlp, HBOOL bAnnotExist=HFALSE)

機能：

現在のページオブジェクトを閉じる

引数：

ctlp ： PDF ファイルのポインタ

bAnnotExist：ページをクローズ後、pl_ClosePdf 間までの間に注釈を追加する必要があるか否かを指定する。説明参照。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PDF 出力シーケンス中の pl_ClosePage でクローズ済みのページに対して、pl_ClosePdf までの間で注釈を追加する場合、bAnnotExist に HTRUE を設定して呼び出す。

8.3.5. カレントページ用紙サイズ設定関数

PDFAPI PL_ERROR pl_SetPagePaper(hPDF ctlp,float pw,float ph)

機能：

現在のページの用紙サイズを設定する。この関数を使用しない場合、現在のページの用紙サイズは PDF ファイルオープン関数で指定されたデフォルト値により決定される。この関数は pl_CreatePage の後、ページ内のデータを出力する前に呼び出す必要がある。

引数：

ctlp ： PDF ファイルのポインタ

pw,ph ： 用紙の幅と高さ。単位はデフォルトユーザ空間の単位である 1/72 インチである。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.3.6. クロップボックスオフセット設定関数

PDFAPI PL_ERROR pl_SetCropOffset(hPDF ctlp,float top,float right ,float bottom,float left, PL_BoxObjE boxObj=PL_BOX_PAGE)

機能：

カレントページ、またはカレントページツリーにクロップボックスを設定する。

引数：

ctlp ： PDF ファイルのポインタ

top、right、bottom、left ： 指定値だけ用紙サイズを拡大して、CropBox に設定する。

boxObj:対象を指定する. PL_BoxObjE の定義は下記通り：

```
typedef enum {
        PL_BOX_PAGE=0,                  // 現在の Page オブジェクト
        PL_BOX_PAGES                    // 現在の Pages オブジェクト
} PL_BoxObjE;
```

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明 ：

pl_SetPagePaper だけが使用された場合、MediaBox、CropBox に指定されたサイズが設定される。pl_SetCropOffset が指定された場合、pl_SetPagePaper で指定されたサイズは TrimBox、ArtBox、BleedBox に設定され、pl_CropOffset で指定された値を外側に拡張した境界ボックスが MediaBox、CropBox に設定される。CropOffset に負の値が設定された場合、0 として処理される。

あわせて pl_SetBleedOffset が指定された場合、pl_SetPagePaper で指定された値が TrimBox、ArtBox に設定され、pl_SetBleedOffset で指定された値を外側に拡張した境界ボックスが BleedBox に設定される。pl_CropOffset が指定されていない場合、MediaBox、CropBox も BleedBox と同じものとなる。BleedOffset が負の値の場合、0 として処理される。また、pl_SetCropOffset が同時に指定された場合、その値は、MediaBox、CropBox に設定されるが、これが BleedBox より小さい場

合は、BleedBox を囲むように拡張される。

8.3.7. ブリードボックスオフセット設定関数

PDFAPI PL_ERROR pl_SetBleedOffset(hPDF ctlp,float top,float right ,float bottom,float left PL_BoxObjE boxObj=PL_BOX_PAGE)

機能：

現在のページ、またはカレントページツリーにブリードボックスを設定する。

引数：

ctlp ： PDF ファイルのポインタ

top、right、bottom、left ： 指定値だけ用紙サイズを拡大して、BleedBox に設定する。

boxObj:対象を指定する。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明 ：

PL_SetCropOffset の説明参照。

8.3.8. トリムボックスオフセット設定関数

PDFAPI PL_ERROR pl_SetTrimOffset(hPDF ctlp,float top,float right ,float bottom,float left, PL_BoxObjE boxObj=PL_BOX_PAGE)

機能：

現在のページ、またはカレントページツリーにトリムボックスを設定する。

引数：

ctlp ： PDF ファイルのポインタ

top、right、bottom、left ： 指定値だけ用紙サイズを拡大して、TrimBox に設定する。

boxObj:対象を指定する。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明 ：

PL_SetCropOffset の説明参照。

8.3.9. アートボックスオフセット設定関数

PDFAPI PL_ERROR pl_SetArtOffset(hPDF ctlp,float top,float right ,float bottom,float left, PL_BoxObjE boxObj=PL_BOX_PAGE)

機能：

現在のページ、またはカレントページツリーにアートボックスを設定する。

引数：

ctlp ： PDF ファイルのポインタ

top、right、bottom、left ： 指定値だけ用紙サイズを拡大して、ArtBox に設定する。

BoxObj:対象を指定する。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明 ：

PL_SetCropOffset の説明参照。

8.3.10. 境界ボックス設定関数

PDFAPI PL_ERROR pl_SetBoundingBox(hPDF ctlp,float x, float y, float pw, float ph,PL_PageBoxModeE pgBoxMode = PL_PGBM_MEDIA)

機能：

現在のページ、またはカレントページツリーに境界ボックスを設定する。

引数：

ctlp ： PDF ファイルのポインタ

x、y ： 開始座標。

pw、ph ： 高さ、幅。

pgBoxMode ： ボックス種別。

```
typedef enum {
        PL_PGBM_MEDIA = 0,      // media box
        PL_PGBM_CROP,           // crop box
        PL_PGBM_BLEED,          // bleed box
        PL_PGBM_TRIM,           // trim box
        PL_PGBM_ART,            // art box
        PL_PGBM_MaxNum          // max number of box modes
} PL_PageBoxModeE;
```

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.3.11. ページ逆順出力設定関数

PDFAPI PL_ERROR pl_SetPageOrder(hPDF ctlp, HBOOL bReverse)

機能：

ページ逆順出力を指定する。

引数：

ctlp ： PDF ファイルのポインタ

bReverse ： 逆順出力するか否かを指定。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.3.12. ページ逆順出力設定関数（ページ範囲指定）

PDFAPI PL_ERROR pl_SetReversePage(hPDF ctlp, int from, int to)

機能：

ページ逆順出力をページ範囲で指定する。

引数：

ctlp ： PDF ファイルのポインタ

from ： 開始ページ。

to ： 終了ページ。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.4. グラフィックス状態設定関数

8.4.1. グラフィックス状態退避関数

PDFAPI PL_ERROR pl_PushState(hPDF ctlp)

機能：

現在のグラフィックス状態を保存する。グラフィックス状態はテキスト等にも適用される。

引数：

ctlp ： PDF ファイルのポインタ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.4.2. グラフィックス状態復旧関数

PDFAPI PL_ERROR pl_PopState(hPDF ctlp)

機能：

前回保存したグラフィックス状態を復旧する。pl_PushState()関数と組み合わせて、pl_PushState()関数の後に使用すること。

引数：

ctlp ： PDF ファイルのポインタ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.4.3. 変換行列設定関数

PDFAPI PL_ERROR pl_SetCm(hPDF ctlp,float a,float b,float c,float d,float e,float f)

機能：

座標の変換行列を設定する。この変換行列はあらゆる出力に影響する。この関数を使用する前に必ず

pl_PushState で現在の状態を保存し、その後 pl_PopState で状態を回復する必要がある(この変換行列の使い方については PDF 仕様書を参照)。

引数：

ctlp ： PDF ファイルのポインタ

a、b、c、d、e、f ： 変換座標の行列要素

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

pl_SetCm()関数で動作する元の座標系はカレントの座標系である。この関数で指定した行列が、現在の変換行列に連結される。

8.4.4. 線幅設定関数

PDFAPI PL_ERROR pl_SetLineWidth(hPDF ctlp,float w)

機能：

線の幅をグラフィックス状態に設定する。

引数：

ctlp ： PDF ファイルのポインタ

w ： 線の幅（ユーザ空間の単位)

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.4.5. ラインキャップスタイル設定関数

PDFAPI PL_ERROR pl_SetLineCap(hPDF ctlp, PL_LineCapE CapStyle)

機能：

開いたサブパスのストローク時に、線の端点に使用される形状（ラインキャップスタイル）をグラフィックス状態に設定する。

引数：

ctlp ： PDF ファイルのポインタ

CapStyle： 線の端点の種類を設定する。デフォルト値は BUTT である。PL_LineCap の定義を以下に示す。

```
typedef enum{
    PL_LC_BUTT=0,        // butt end caps
    PL_LC_ROUNDC,        // round end caps
    PL_LC_SQUARE         // projecting square end caps
}PL_LineCapE;
```

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：下記に CapStyle の例を示す。

種別	形状	説明
PL_LC_BUTT		パスの端点でストロークを角型に打ち切る。
PL_LC_RUNDOC		線幅に等しい直径の半円弧を端点の周囲に描き、内部を塗りつぶす。
PL_LC_SQUARE		パスの端点を越えて、線幅の半分の距離までストロークを続け、角型に打ち切る。

表 8.4.5 ラインキャップスタイル

8.4.6. ラインジョインスタイル設定関数

PDFAPI PL_ERROR pl_SetLineJoin(hPDF ctlp, PL_LineJoinE JoinStyle)

機能：

パス内で連続する線分の接続箇所で使用される形状（ラインジョインスタイル）をグラフィックス状態に設定する。

引数：

ctlp ： PDF ファイルのポインタ

JoinStyle ：パスの連続部分の形状を設定する。

PL_LineJoinE の定義を以下に示す。

```
typedef enum{
    PL_LJ_MITER=0,      // miter joins
    PL_LJ_ROUND,        // round joins
    PL_LJ_BEVEL         // bevel joins
}PL_LineJoinE;
```

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

下記に JoinStyle の例を示す

種別	形状	説明
PL_LJ_MITER		２つの線分のストロークの外縁部を、絵画の額縁のように、ある角度で交わるまで延長する。

PL_LJ_ROUND		直径が線幅に等しい円弧を線分の交点を中心として描いて塗りつぶし、角を丸める。
PL_LJ_BEVEL		２つの線分の端点を前項のバットキャップで仕上げ、端点にできた三角形を塗りつぶす。

表 8.4.6 ラインジョインスタイル

8.4.7. マイターリミット設定関数

PDFAPI PL_ERROR pl_SetMiterLimit(hPDF ctlp,float miterlimit)

機能：

二つの線分の接続点のマイターリミットをグラフィックス状態に設定する。

引数：

ctlp ： PDF フィアルのポインタ

miterlimit ： マイターリミット値(ユーザ空間単位)

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

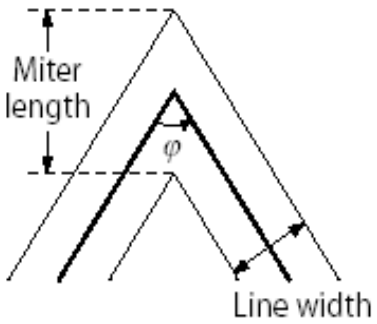
説明：

miterlimit 引数の取得方法は二種類ある。

(1). miterlimit =Miter length/Line width

(2). miterlimit =1/sin(φ/2)

以下に例を示す。



8.4.8. 線パターン設定関数

PDFAPI PL_ERROR pl_SetLineType(hPDF ctlp, PL_LinePatternTD * lp)

機能：

線のパターンをグラフィックス状態に設定する。

引数：

ctlp ： PDF ファイルのポインタ

lp ： 線のパターン構造体へのポインタ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_LinePatternTD 構造体については共通データ説明を参照。

8.4.9. レンダリングインテント設定関数

PDFAPI PL_ERROR pl_SetIntent(hPDF ctlp, PL_RenderingIntentE renderingIntent)

機能：

レンダリングインテントを設定する。

引数：

ctlp ： PDF ファイルのポインタ

renderingIntent： レンダリングインテント

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

レンダリングインテントには下記を設定する。

```
typedef enum {
    PL_RI_Relative = 1,          // RelativeColorimetric,default value.
    PL_RI_Absolute,              // AbsoluteColorimetric
    PL_RI_Saturation,            // Saturation
    PL_RI_Perceptual             // Perceptual
} PL_RenderingIntentE;
```

8.4.10. グラフィックス状態設定関数

PDFAPI PL_ERROR pl_SetGraphState(hPDF ctlp,const PL_GraStateTD& GraState)

機能：

指定されたパラメータをグラフィックス状態に設定する。

引数：

ctlp ： PDF ファイルのポインタ

GraState ： グラフィックス状態パラメータ。詳しくは説明参照

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_GraStateTD：グラフィックス状態パラメータを設定する。定義を以下に示す。

```
typedef struct {
    PL_GSFlagE          gsFlag;  // 設定するパラメータを指定する
    PL_BlendModeE       BM;      // 透過イメージングモデルで使用されるブレンドモード
    float               CA;      // ストローク用α定数 (0.0 から 1.0)
    float               ca;      // 非ストローク用α定数 (0.0 から 1.0)
    HBOOL               bAIS;    // α ソースフラグ
                                 // (HTRUE： αをソースとみなす,
                                 // HFALSE： αを不透明度とみなす).
    HBOOL               bTK;     // テキストノックアウトフラグ
                                 //      HTRUE：全グリフを一つのオブジェクトとみなす。
                                 //      HFALSE：各グリフを独立した要素とみなす。
    HBOOL               bOP;     // オーバプリントモード(ストローク用)可否
    HBOOL               bop;     // オーバプリントモード(塗りつぶし用)可否
    PL_OPModeE          OPM;     // オーバプリントモード
    PL_RenderingIntentE RI;      // レンダリングインテント
    float               FL;      // 平滑許容度(正の値)
    float               SM;      // 円滑許容度(0.0 から 1.0)
    HBOOL               bSA;     // 自動ストローク調整の適用可否
} PL_GraStateTD;
```

gsFlag の定義を以下に示す。設定する値に対応するビットをセットする。

```
typedef enum {
 PL_GSF_NONE    = 0,
 PL_GSF_BM      = 1 << 0,       // BM 指定有
 PL_GSF_CA      = 1 << 1,       // CA 指定有
 PL_GSF_ca      = 1 << 2,       // ca 指定有
 PL_GSF_AIS     = 1 << 3,       // AIS 指定有
 PL_GSF_TK      = 1 << 4,       // TK 指定有
 PL_GSF_OP      = 1 << 5,       // OP 指定有
 PL_GSF_op      = 1 << 6,       // op 指定有
 PL_GSF_OPM     = 1 << 7,       // OPM 指定有
 PL_GSF_RI      = 1 << 8,       // RI 指定有
 PL_GSF_FL      = 1 << 9,       // FL 指定有
 PL_GSF_SM      = 1 << 10,      // SM 指定有
 PL_GSF_SA      = 1 << 11       // SA 指定有
} PL_GSFlagE;
```

PL_BlendModeE の定義を以下に示す。

typedef enum{

PL_BM_NORMAL=0,PL_BM_MULTIPLY,PL_BM_SCREEN,PL_BM_OVERLAY,

PL_BM_DARKEN,PL_BM_LIGHTEN,PL_BM_COLORDODGE,

PL_BM_COLORBURN,PL_BM_HARDLIGHT,PL_BM_SOFTLIGHT,

PL_BM_DIFFERENCE,PL_BM_EXCLUSION,PL_BM_HUE,

PL_BM_SATURATION,PL_BM_COLOR,PL_BM_LUMINOSITY

} PL_BlendModeE;

PL_OPModeE の定義を以下に示す。

typedef enum {

PL_OPM_ZERO = 0, // 0 オーバープリントモード

PL_OPM_NONZERO // 非 0 オーバープリントモード

} PL_OPModeE;

各値についての詳細は PDF Reference 参照

PDF/X 出力時、ストローク用、非ストローク用の α 定数は 1 でなければならない。透明画像イメージングモデルに使用されるブレンドモードは Normal でなければならない。

8.4.11. カラー設定関数

PDFAPI PL_ERROR pl_SetColor(hPDF ctlp, PL_OperatorE mode,float a,float b=0.0,float c=0.0,float d=0.0)

機能：

PDF ではストロークオペレーション用の色と非ストロークオペレーション（塗りつぶし）用の色が存在する。このいずれかを選択し、カラー値を設定する。カラースペースは事前に pl_SetColorSpace(後述)で設定しておく必要がある。ただし、デバイスカラースペースの場合は、pl_SetColorSpace による設定を省略して、直接、この関数で指定することもできる。

引数：

ctlp ： PDF ファイルのポインタ

mode ： 色のパターン。以下のいずれかを設定する

PL_I_g カラースペースを DeviceGray に設定し、塗りつぶし色を設定する

PL_I_G カラースペースを DeiviceGray に設定し、ストローク色を設定する

PL_I_k カラースペースを DeviceCMYK に設定し、塗りつぶし色を設定する

PL_I_K カラースペースを DeviceCMYK に設定し、ストローク色を設定する

PL_I_rg カラースペースを DeviceRGB に設定し、塗りつぶし色を設定する

PL_I_RG カラースペースを DeviceRGB に設定し、ストローク色を設定する

PL_I_sc 塗りつぶし色を設定する。

PL_I_SC ストローク色を設定する。

a,b,c,d：カラースペースによって、意味が異なる。

デバイスカラースペースの場合、いずれの場合も、0.0～1.0 の範囲の数値で下記のようになる。

PL_I_rg, PL_I_RG の場合、 a：Red 値、b：Green 値、c：Blue 値(d 設定不要)

PL_I_g, PL_I_G の場合、a：Gray 値(b,c,d 設定不要)

PL_I_k, PL_I_K の場合、 a：Cyan 値、b：Magenta 値、c：Yellow 値、d：Black 値

デフォルト値は DeviceGray カラースペースの、a = 0.0(黒)である。

PDF/X:

PDF/X-1:RGB 及び他の Device Independent カラースペースの設定は許可されない。

PDF/X-3：RBG カラースペースを使用する場合、出力デバイスが RGB サポートしない場合、DefaultRGB を設定しなければならない。

同様に、CMYK カラースペースを使用する場合、出力デバイスが CMYK をサポートしない場合、DefaultCMYK を設定しなければならない。

同様に、Gray カラースペースを使用する場合、出力デバイスは Gray 或は CMYK をサポートしない場合、DefaultGray を設定しなければならない。

PDF/A:

ICCBased カラースペースを使用する場合、ICC カラープロファイルを指定する必要がある。DeviceRGB,および DeviceCMYK は使用可能であるが、同時に使用することはできず、また、使用する場合、OutputIntent を指定する必要がある。Separation を使用する場合、その代替カラーは、上記の ICCBased および DeviceRGB,DeviceCMYK の制限に従う必要がある。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.4.12. カラースペースロード関数

PDFAPI PL_ERROR pl_LoadColorSpace(hPDF ctlp,PL_ColorSpaceTD* csp, int& csIdx);

機能：

指定されたカラースペースをロードしカラースペースのインデクスを戻す。

引数：

ctlp ： PDF ファイルのポインタ

csp ： ロードするカラースペースを指定する

csIdx：カラースペースインデクス(戻り)

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

PL_ERR_EmptyPoint

カラースペースの格納用に指定されたポインタ、あるいはセパレーションカラーのカラー名が Null

PL_ERR_CS_UnAllowedColorSpace

許可されないカラースペース指定。色なしパターン用のカラースペースにパターンカラースペースが指定された、あるいは、セパレーションカラースペースの代替カラースペースにパターン、あるいはセパレーションカラーススペースを指定した場合など

PL_ERR_CS_InvalidWhitePoint

CalGray、CalRGB、Lab で、無効なホワイトポイントが指定された。

PL_ERR_CS_InvalidBlackPoint

CalGray、CalRGB、Lab で、無効なブラックポイントが指定された。

PL_ERR_CS_InvalidGamma

CalGray、CalRGB で、無効なガンマ値が指定された。

PL_ERR_CS_InvalidRange

Lab カラースペースで、無効な range が指定された。

PL_ERR_CS_UnsupportedColorSpace

サポートしないカラースペースが指定された（PL_ColorType 定義以外の値）。

説明：

PL_ColorSpaceTD の定義については共通データ形式参照。

csIdx にはカラースペースのインデクスが戻る。これは pl_SetColorSpace()関数などのカラースペースインデクスを必要とする箇所で使用する。

8.4.13. カラースペース設定関数

PDFAPI PL_ERROR pl_SetColorSpace(hPDF ctlp,int csIdx, PL_OperatorE mode)

機能：

カラースペースを設定する。

引数：

ctlp：PDF ファイルポインタ

csIdx：pl_LoadColorSpace で取得したカラースペースインデクス。

mode：色の設定箇所の選択。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

PL_ERR_CS_Empy：ロードされていないカラースペースのような無効値が指定された場合。

説明：

mode モード。下記の 2 種類のいずれかを指定する。

PLI_cs　塗りつぶし用の色

PLI_CS　ストロークの色

8.4.14. カラープロファイルロード関数

PDFAPI PL_CPHandle pl_LoadColorProfile (hPDF ctlp,const char* colorProfile);

機能：

指定されたカラープロファイルをロードしカラープロファイルハンドルを戻す。このハンドルは pl_LoadColorSpace、pl_SetImageColorProfileOpt などで使用する。

引数：

ctlp ： PDF ファイルのポインタ

colorProfile ： プロファイルのパス

戻り値：

正常終了の場合、カラープロファイルのハンドルが戻る。それ以外の場合、NULL が戻る

8.4.15. カラープロファイルロード関数(ストリーム)

PDFAPI PL_CPHandle pl_LoadColorProfile_Stream (hPDF ctlp,std::istream& ismColorProfile);

機能：

カラープロファイルを指定されたストリームからロードし、カラープロファイルハンドルを戻す。このハンドルは pl_LoadColorSpace、pl_SetImageColorProfileOpt などで使用する。

引数：

ctlp ： PDF ファイルのポインタ

ismColorProfile ： プロファイルストリーム

戻り値：

正常終了の場合、カラープロファイルのハンドルが戻る。それ以外の場合、NULL が戻る

8.4.16. カラープロファイルロード関数(メモリバッファエリア)

PDFAPI PL_CPHandle pl_LoadColorProfile_Buffer (hPDF ctlp,std::string& cBufColorProfile);

機能：

カラープロファイルを指定されたバッファエリアからロードし、カラープロファイルハンドルを戻す。このハンドルは pl_LoadColorSpace、pl_SetImageColorProfileOpt などで使用する。

引数：

ctlp ： PDF ファイルのポインタ

cBufColorProfile ： プロファイルバッファエリア

戻り値：

正常終了の場合、カラープロファイルのハンドルが戻る。それ以外の場合、NULL が戻る

8.4.17. デフォルトカラースペース設定関数

PDFAPI PL_ERROR pl_SetDefaultColorSpace(hPDF ctlp,PL_DefaultColorTypeE defaultColorType, int defaultCSIdx)

機能：

デフォルトカラースペースを設定する。

引数：

ctlp：PDF ファイルポインタ

defaultColorType：デフォルトカラースペースのタイプを指定する。定義を以下に示す。

```
typedef enum {
        PL_CS_DEFAULTGRAY = 0,          // デフォルトグレイを設定する
        PL_CS_DEFAULTRGB,               // デフォルト RGB を設定する
```

PL_CS_DEFAULTCMYK　　// デフォルト CMYK を設定する

} PL_DefaultColorTypeE;

defaultCSIdx：デフォルトカラースペースとして割り当てるカラースペースのインデクス。pl_LoadColorSpace で取得する。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

PL_ERR_CS_Empy：ロードされていないカラースペースのような無効値が指定された場合。

PL_ERR_CS_UnAllowedColorSpace：Lab,および Pattern などデフォルトカラースペースで指定できないカラースペースが割り当てられた場合。あるいは、カラー成分が 1 つである DefaultGray に対して、DeviceRGB を割り当てた場合などに発生する。

説明：

PDF のデフォルトカラースペースは各ページ単位で割り当てるものである。

PDF/X-1 ではデフォルトカラースペースは設定できない。

8.4.18.　カラープロファイル検査関数

PDFAPI PL_ERROR pl_CheckColorProfile (hPDF ctlp,const char* colorProfile);

機能：

指定されたカラープロファイルが出力中の PDF で使用可能か否かを確認する。

引数：

ctlp ： PDF ファイルのポインタ

colorProfile ： プロファイルのパス

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

PL_ERR_CS_UnsupportedICCVer

サポートされていないバージョンの icc color profile である。PDF 各バージョンがサポートする icc color profile のバージョンを以下に示す。

PDF1.3：ICC バージョン 3.3 でサポートする

PDF1.4：1998-09 版までサポートする

PDF1.5 以降：2001-12(4.0)版までサポートする

PL_ERR_CS_UnsupportedICCDevice

サポートされていないデバイスクラス用の icc color profile である。PDF でサポートされるデバイスクラスを以下に示す。

icSigInputClass ('scnr'), icSigDisplayClass ('mntr'),

icSigOutputClass ('prtr'), icSigColorSpaceClass ('spac')

PL_ERR_CS_UnsupportedICCColor

サポートされないカラースペース用の icc color profile である。PDF でサポートされるカラースペースを以下に示す。

icSigGrayData ('GRAY'), icSigRgbData ('RGB ')'

icSigCmykData ('CMYK'), icSigLabData ('Lab ')

8.4.19. カラープロファイル検査関数(ストリーム)

PDFAPI PL_ERROR pl_CheckColorProfile_Stream (hPDF ctlp,std::istream& ismColorProfile);

機能：

ストリームに指定されたカラープロファイルが出力中の PDF で使用可能か否かを確認する。

引数：

ctlp ： PDF ファイルのポインタ

ismColorProfile ： プロファイルのストリーム

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.4.20. カラープロファイル検査関数(メモリバッファエリア)

PDFAPI PL_ERROR pl_CheckColorProfile_Buffer (hPDF ctlp,std::string& strColorProfile);

機能：

メモリバッファに指定されたカラープロファイルが出力中の PDF で使用可能か否かを確認する。

引数：

ctlp ： PDF ファイルのポインタ

strColorProfile ： プロファイルのメモリバッファエリア

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.4.21. カラープロファイル情報取得関数

PDFAPI PL_ERROR pl_GetColorProfileInformation (hPDF ctlp, PL_CPHandle hColorProfile,
PL_CPInfoTypeE cpInfoType,PL_ColorProfileInfoTD& cpInfo)

機能：

カラープロファイルの情報を取得する。

引数：

ctlp ： PDF ファイルポインタ

hColorProfile：カラープロファイルのハンドル(pl_LoadColorProfile で取得する)

cpInfoType：取得する情報を指定する

cpInfo：取得したカラープロファイル情報

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

cpInfoType：必要な情報を指定する。定義を以下に示す。

```
typedef enum {
    PL_CPI_VER          = 1 << 0,       // バージョン
    PL_CPI_DEVICE       = 1 << 1,       // 取得する
    PL_CPI_COLORSPACE   = 1 << 2,       // カラースペース
    PL_CPI_CHANNELS     = 1 << 3,       // チャネル
    PL_CPI_RANGE        = 1 << 4,       // レンジ
    PL_CPI_ALL =
        PL_CPI_VER|PL_CPI_DEVICE|PL_CPI_COLORSPACE|PL_CPI_CHANNELS|
        PL_CPI_RANGE
} PL_CPInfoTypeE;
```

cpInfo にカラープロファイルの各種情報を取得する。PL_ColorProfileInfoTD の定義を以下に示す。

```
typedef struct {
    PL_CPVersionE   cpVer;          // プロファイルバージョン
    PL_CPDeviceE    cpDevice;       // デバイスクラス
    PL_CPColorSpaceE cpColorSpace;  // カラースペース
    int             cpChannels;     // カラー成分数
    float           cpRange[8];     // 成分毎の値の範囲(cpChanel*2 個を使用)
} PL_ColorProfileInfoTD;
```

cpVer はカラープロファイルの仕様で定義されている形式となる。

```
typedef enum {
    PL_CPV_NONE = 0,
    PL_CPV_33     = 0x02100000,     //(PDF1.3 以降で使用可能)
    PL_CPV_1998 = 0x02200000,       //(PDF1.4 以降で使用可能)
    PL_CPV_2001 = 0x04000000,       //(PDF1.5 以降で使用可能)
    PL_CPV_2003 = 0x04000000        //(PDF1.5 以降で使用可能)
} PL_CPVersionE;
```

PL_CPDeviceE の定義を以下に示す。

```
typedef enum {
    PL_CP_InputClass = 0x73636E72L,         //  'scnr'
    PL_CP_DisplayClass = 0x6D6E7472L,       //  'mntr'
    PL_CP_OutputClass = 0x70727472L,        //  'prtr'
    PL_CP_LinkClass = 0x6C696E6BL,          //  'link'
    PL_CP_AbstractClass = 0x61627374L,      //  'abst'
    PL_CP_ColorSpaceClass = 0x73706163L,    //  'spac'
    PL_CP_NamedColorClass = 0x6e6d636cL,    //  'nmcl'
    PL_CP_MaxEnumClass = 0xFFFFFFFFL
} PL_CPDeviceE;
```

PDF で使用可能なデバイスは、'scnr' 、'mnt ' 、'prtr' 、'spac'の 4 種類である。

PL_CPColorSpaceE の定義を以下に示す。

```
typedef enum {
        PL_CP_XYZData= 0x58595A20L,             // 'XYZ '
        PL_CP_LabData= 0x4C616220L,             // 'Lab '
        PL_CP_LuvData= 0x4C757620L,             // 'Luv '
        PL_CP_YCbCrData= 0x59436272L,           // 'YCbr'
        PL_CP_YxyData = 0x59787920L,            // 'Yxy '
        PL_CP_RgbData = 0x52474220L,            // 'RGB '
        PL_CP_GrayData = 0x47524159L,           // 'GRAY'
        PL_CP_HsvData = 0x48535620L,            // 'HSV '
        PL_CP_HlsData = 0x484C5320L,            // 'HLS '
        PL_CP_CmykData = 0x434D594BL,           // 'CMYK'
        PL_CP_CmyData    = 0x434D5920L,         // 'CMY '
        PL_CP_2colorData = 0x32434C52L,         // '2CLR'
        PL_CP_3colorData = 0x33434C52L,         // '3CLR'
        PL_CP_4colorData = 0x34434C52L,         // '4CLR'
        PL_CP_5colorData = 0x35434C52L,         // '5CLR'
        PL_CP_6colorData = 0x36434C52L,         // '6CLR'
        PL_CP_7colorData = 0x37434C52L,         // '7CLR'
        PL_CP_8colorData = 0x38434C52L,         // '8CLR'
        PL_CP_9colorData = 0x39434C52L,         // '9CLR'
        PL_CP_10colorData = 0x41434C52L,        // 'ACLR'
        PL_CP_11colorData = 0x42434C52L,        // 'BCLR'
        PL_CP_12colorData = 0x43434C52L,        // 'CCLR'
        PL_CP_13colorData = 0x44434C52L,        // 'DCLR'
        PL_CP_14colorData = 0x45434C52L,        // 'ECLR'
        PL_CP_15colorData = 0x46434C52L,        // 'FCLR'
        PL_CP_MaxEnumData = 0xFFFFFFFFL
} PL_CPColorSpaceE;
```

PDF でサポートされるカラースペースは、"Gray" 、"RGB " 、"CMYK" 、"Lab "である。

cpChannels はカラー成分の数であり、PDF でサポートされるのは、１、３、４である。

cpRange は各カラー成分の値の下限、上限を示すものであり、第一成分の下限、上限、第二成分の下限、上限、... の順となる。

8.5. パス関連オペレータ出力関数

PDF 内の線画関連のグラフィックスは、複数の点の間を接続する「パス」を作成し、最後にこれをペイントする、という操作により描画される。「パス」は連続した 1 本の線分でなくてもよい。この場合、不連続なそれぞれの線分は「サブパス」と呼ばれる。作成中のサブパスの終点はカレントポイントと呼ばれる。本項の各種のパス設定関数群は、カレントポイントに接続する新しい頂点を指定して、サブパスを延長していくものである。サブパスの作成は描画動作を伴うものでなはいため、ペイント関数を使用するまでは描画操作は行われない。

本ライブラリで、サブパスの起点（それまでの点と接続されていない点）を設定する場合、カレントポイント設定関数を使用する。以降、各種のパス出力関数を使用して、この点からの線分を接続していくことになる。

カレントポイント設定関数のほかに、矩形パス出力関数、円パス出力関数もサブパスの起点を作成する。パスの作成中にこの関数を使用した場合、作成中のサブパスは終了となり、これにより作成される矩形パス、円パスは、新しいサブパスとなる。

なお、設定されたパスはクリッピングパスとして使用することもできる。設定したパスをクリッピングパスにする場合はクリッピングパス設定関数を使用する。

8.5.1. カレントポイント設定関数

PDFAPI PL_ERROR pl_GraphicTo(hPDF ctlp,float a,float b)

機能：

線画の位置を設定する。指定された点は新しいサブパスの開始点となり、同時にこのサブパスのカレントポイントとなる。

引数：

ctlp：PDF ファイルのポインタ

a,b ：新しいカレントポイントとする位置の座標(ユーザ空間)

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.5.2. 直線パス出力関数

PDFAPI PL_ERROR pl_LineTo(hPDF ctlp,float a,float b)

機能：

カレントポイントから指定された点までの直線パスを出力する。指定された点がパスの新しいカレントポイントとなる。

引数：

ctlp：PDF ファイルのポインタ

a,b： 指定する点の位置座標(x,y) (ユーザ空間)

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.5.3. 3 次ベジェ曲線パス出力関数 1

PDFAPI PL_ERROR pl_CurveTo(hPDF ctlp,float a,float b,float c,float d,float e,float f)

機能：

カレントポイントから指定された点にベジェ曲線パスを出力する。パラメータで２つの制御点を指定する。呼び出し時のカレントポイントがベジェ曲線の開始点であり、終点がパスの新しいカレントポイントとなる。

引数：

ctlp：PDF ファイルのポインタ

a,b,c,d：ベジェ曲線の制御点

e,f：ベジェ曲線の終点

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.5.4. 3 次ベジェ曲線パス出力関数 2

PDFAPI PL_ERROR pl_CurveTo_v(hPDF ctlp,float a,float b,float c,float d)

機能：

現在の点から指定された点にベジェ曲線パスを出力する。呼び出し時のカレントポイントがベジェ曲線の開始点であり、制御点も兼ねる。終点がパスの新しいカレントポイントとなる。

引数：

ctlp：PDF ファイルのポインタ

a,b：ベジェ曲線の制御点座標

c,d：ベジェ曲線の終点座標

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.5.5. 3 次ベジェ曲線パス出力関数 3

PDFAPI PL_ERROR pl_CurveTo_y(hPDF ctlp,float a,float b,float c,float d)

機能：

現在の点から指定された点にベジェ曲線パスを出力する。パラメータで指定された点と終点が制御点となる。終点がパスの新しいカレントポイントとなる。

引数：

ctlp ： PDF ファイルのポインタ

a,b ： ベジェ曲線の制御点

c,d ： ベジェ曲線の終点座標（制御点を兼ねる）

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.5.6. 円パス出力関数

PDFAPI PL_ERROR pl_Circle(hPDF ctlp,float x,float y,float r)

機能：

サークルの中心と半径を指定して、円形のパスを出力する。指定された円形のパスは開いた状態となっているため、必要であればパスクローズ関数を使用して、パスを閉じる必要がある。カレントポイントは(x、y+r) となる。

引数：

ctlp ： PDF ファイルのポインタ

x,y ： サークルの中心の座標

r ： サークルの半径

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PDF には円に相当するパス出力は存在しない。このため、前項までの各パス出力関数と異なり、PDF のオペレータと直接対応するものでなない。本関数はベジェ曲線 4 本を使用して円パスを作成する。

8.5.7. パスクローズ関数

PDFAPI PL_ERROR pl_ClosePath(hPDF ctlp)

機能：

現在のパスを閉じる（現在サブパスのカレントポイントからサブパスの開始点に直線を追加してパスを閉じる)。

引数：

ctlp ： PDF ファイルのポインタ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.5.8. 矩形パス出力関数

PDFAPI PL_ERROR pl_Rect(hPDF ctlp,float a,float b,float c,float d)

機能：

指定された位置に、指定の幅と高さの矩形のサブパスを出力する。指定した矩形のパスは閉じられた状態となる。カレントポイントは(a,b)で指定される点となる。

引数：

ctlp ： PDF ファイルのポインタ

a,b ： 矩形の左下角隅点の座標

c,d ： 矩形の幅と高さ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.5.9. ペイント関数

PDFAPI PL_ERROR pl_Paint(hPDF ctlp, PL_OperatorE mode, unsigned char transparency = 0)

機能：

ペイントとは、ストローク、塗りつぶし、あるいはその双方を行うことである（PDF 中のすべての線画に関するコマンドは pl_Paint 関数を使って描画する必要がある)。

引数：

ctlp ： PDF ファイルのポインタ

mode ：ペイント方法の指定。

transparency ：透明度（%）範囲 0～100。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

mode には以下のいずれかのモードを設定する。

PL_I_f	非ゼロ回転数規則で現在のパスを塗りつぶす
PL_I_f8	奇偶規則で現在のパスを塗りつぶす
PL_I_n	現在の線画パスを終了する（ストローク、塗りつぶし無し）
PL_I_s	現在の線画パスを閉じ、ストロークする
PL_I_S	現在の線画パスをストロークする
PL_I_b	現在の線画パスを閉じ非ゼロ回転数規則で塗りつぶしストロークする
PL_I_b8	現在の線画パスを閉じ奇偶規則で塗りつぶし、ストロークする
PL_I_B	非ゼロ回転数規則で現在の線画パスを塗りつぶし、ストロークする
PL_I_B8	奇偶規則で現在の線画パスを塗りつぶし、ストロークする

transparency は透過を実現できない PDF バージョン（PDF1.3）で、透過に近い表現を疑似的に表すためのオプションであるため、そのような出力を行わない場合は 0（ゼロ）を指定するか省略する。

8.5.10. クリッピングパス設定関数

PDFAPI PL_ERROR pl_PathClip(hPDF ctlp,PL_PointTD* ptp,int n, PL_OperatorE ClipMode)

機能：

指定されたパスに従ってクリッピングパスを設定する。

引数：

ctlp ： PDF ファイルのポインタ

ptp ： 指定するパスの一系列の点。構造体については説明参照。

n ： 指定するパスの点数

ClipMode ： 以下のいずれかの計算規則を指定してクリップする。

PL_I_W 非ゼロ回転数規則を使用する

PL_I_W8 奇偶規則を使用する

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_PointTD 構造体の定義については共通データ形式参照

通常は、指定された一連の点列により、クリップパスを作成し、h オペレータでパスを閉じた後、 W または W* オペレータを出力する。ただし、n=0 の場合、W または W* を出力する。これは前項までの各種パス設定関数で作成したパスをクリッピングパスに設定するためのものである。

8.6. テキストオペレータ出力関数

8.6.1. フォント設定関数

PDFAPI PL_ERROR pl_SetFont(hPDF ctlp,PL_FontTD* wfp)

機能：

次のテキストの出力に使用するフォントを設定する。

引数：

ctlp ： PDF ファイルのポインタ

wfp ：フォントの指定

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_FontTD 構造体の定義を以下に示す。

```
typedef struct {
        UCHAR_t         UFName[FS_NAMELEN];         // フォント名
        PL_FontPropE    flag;                       // 属性
```

```
    int         Weight;                         // ボールド属性
    float       size;                           // 文字サイズ
    PL_ColorTD  color;                          // 文字色
    PL_LineTD   strikeline,underline,overline;  // ラインの特性
} PL_FontTD;
```

UFName： システムエンディアンの Unicode でフォントの名称を指定する。

flag ：フォントの属性を指定する。

```
typedef enum {
    PL_FP_NONE          = 0,
    PL_FP_ITALIC        = 1 << 1,       // 斜体字
    PL_FP_UNDERLINE     = 1 << 2,       // 下線
    PL_FP_OVERLINE      = 1 << 3,       // 上線
    PL_FP_STRIKELINE    = 1 << 4,       // 取消線
    PL_FP_SUPERSCRIPT   = 1 << 5,       // 上付き文字
    PL_FP_SUBSCRIPT     = 1 << 6,       // 下付き文字
    PL_FP_NOTUSECOLOR   = 1 << 7,       // color 内で設定した色を無視
    PL_FP_OBLIQUE       = 1 << 8,       // 斜め文字
    PL_FP_BACKSLANT     = 1 << 9        // 後ろへの傾け
} PL_FontPropE;
```

Weight：フォントのウェイト属性を指定する。ウェイト値は 100 から 900 までの 100 単位の値であり、通常、Regular 書体は 400、Bold 書体は 700 を持つ。

size： フォントのサイズを指定する。

color： フォントの色を指定する。

strikeline,underline,overline：取消線、下線、上線のスタイルと色を指定する。

PL_LineTD 構造体の定義については共通データ形式参照。

PDF/X、PDF/A では全てのフォントを埋め込む必要がある。

また、PDF/X、PDF/A の色についての制限にも従う必要がある。

8.6.2. 記述方向設定関数

PDFAPI PL_ERROR pl_SetWriteDirect(hPDF ctlp, PL_TxtWrtModeE wMode, HBOOL Reverse)

機能：

文字の記述方向を設定する。本関数は pl_SetFont()関数でフォントを設定した後で、使用すること。

引数：

ctlp ： PDF ファイルのポインタ

wMode ： 出力方向を設定する。この変数の定義を以下に示す。

```
typedef enum{
    PL_D_HORIZONTAL=0,PL_D_VERTICAL1,PL_D_VERTICAL2
} PL_TxtWrtModeE;
```

PL_D_HORIZONTAL　横書き（デフォルト）

PL_D_VERTICAL1　縦書き

PL_D_VERTICAL2　縦書き(横書きから時計周りの順で 90 度回転する)

Reverse ： テキストの上下方向をページ座標系と反転させる。

HFALSE 反転しない（デフォルト値）

HTRUE 反転する（元のテキストを 180 度回転する）

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.6.3. 変換行列設定関数

PDFAPI PL_ERROR pl_SetTm(hPDF ctlp,float a,float b,float c,float d,float e,float f)

機能：

座標の変換行列を設定する。この変換行列はテキスト出力だけに影響する。この関数を使う前に必ず pl_PushState でカレント状態を保存し、pl_PopState で状態を回復できるようにしておく必要がある。

引数：

ctlp ： PDF ファイルのポインタ

a、b、c、d、e、f ： 座標の行列変換の要素

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

pl_SetCm 同様に、引数の a、b、c、d、e、f は行列要素である。ただし、pl_SetCm()関数と異なり、動作の対象となる座標系はデフォルト座標系である。現在使用している座標系ではない。この関数は常に[1,0,0,1,0,0]の座標系に対して作用する。

8.6.4. テキスト状態パラメータ設定関数

PDFAPI PL_ERROR pl_SetTextParas(hPDF ctlp, PL_OperatorE Ins,float f1)

機能：

テキスト状態を示すパラメータを設定する。

引数：

ctlp ： PDF ファイルのポインタ

Ins ： f1 に設定される引数の意味を示す。詳しくは説明参照

f1 ： 引数の値

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

Ins　下記のいずれかのパラメータを指定する

PL_I_Tc：　文字スペーシング (0)

PL_I_TL : 行のインテル値。同時に行の高さを設定する (0)

PL_I_Tr : テキストのレンダリングモード、下図参照(0)

PL_I_Ts : テキストライズ (0)

PL_I_Tw : 単語スペーシング(スペースのサイズにも影響する) (0)

PL_I_Tz : 水平スケーリング(100)

括弧内はデフォルト値を示す。

レンダリングモード値	例	説明
0		テキストを塗りつぶし
1		テキストをストローク
2		テキストを塗りつぶし、次にストローク
3		テキストを塗りつぶしもストロークもしない
4		テキストを塗りつぶし、パスに追加してクリップ
5		テキストをストロークし、パスに追加してクリップ
6		テキストを塗りつぶし、次にストロークし、パスに追加してクリップ
7		テキストをパスに追加してクリップ

表 8.6.4 テキストレンダリングモード

8.6.5. 出力位置設定関数

PDFAPI PL_ERROR pl_TextTo(hPDF ctlp,float dx,float dy)

機能：

テキストを出力する位置を設定する。テキストを出力する前にこの関数で位置決めを行う。同じ行に連続してテキストを出力する場合は、この関数を使わなくてもかまわない。

引数：

ctlp： PDF ファイルのポインタ

dx,dy：座標(PDF 座標系、rtf 座標 いずれも指定可)

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.6.6. 改行出力関数

PDFAPI PL_ERROR pl_NextLine(hPDF ctlp)

機能：

改行を行う。継続するテキスト出力を次の行に対するものとする。

引数：

ctlp ： PDF ファイルのポインタ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明:

現在、本関数は未サポートである。

8.6.7. 文字列出力関数

PDFAPI PL_ERROR pl_ShowTextU(hPDF ctlp, const UCHAR_t* ustrp)

機能：

現在の位置に Unicode 文字列を出力する。

引数：

ctlp ： PDF ファイルのポインタ

ustrp ： システムエンディアンと一致する Unicode で出力する文字列を指定する。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.6.8. 文字列出力関数 2

PDFAPI PL_ERROR pl_ShowTextU2(hPDF ctlp, const UCHAR_t* lpUStr1, const UCHAR_t* lpUStr2, const PL_StrMapTD& UStrMap)

機能：

カレント位置に Unicode 文字列を出力する。

引数：

ctlp：PDF ファイルのポインタ

lpUStr1：システムエンディアンと一致する Unicode で lpUStr2 の元の文字列を指定する。

lpUStr2：システムエンディアンと一致する Unicode で出力する文字列を指定する。

UStrMap:lpUStr1 と lpUStr2 の対応付けを示す。 StrMapTD の定義は共通データ形式を参照のこと。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明:

OpenTypen の Glyph Substitution 相当の置換機能を Unicode のコードポイント上に別の文字を定義することで実装したフォント（タイ語対応フォントに見られる）の出力を想定した機能である。

lpUStr2 に指定された Unicode を PDF に出力する。lpUStr1 に指定されるオリジナル Unicode は、PDF 内に Unicode とグリフ番号の対応付け（ToUnicode CMap）を作成するために使用する（これは Acrobat 上で文字の検索などで使用される)。

PL_StrMapTD

変換元の Unicode 文字列と変換先の Unicode 文字列(あるいはグリフ番号列)の対応関係を設定する。定義を以下に示す。

```
typedef struct {
        int             StrNum; //   lpIdx の長さ
        PL_IdxTD        *lpIdx;    // 元の Unicode 文字列と変換後の Unicode 文字列
                                   //    (あるいはグリフ番号列)の対応関係のテーブル
} PL_StrMapTD;
```

PL_IdxTD の定義を以下に示す

```
typedef struct {
        int SrcPos;     // 元の Unicode 文字列中の位置
        int DstPos;     // 変換後の Unicode(あるいはグリフ番号)の位置
} PL_IdxTD;
```

UsrtMap の設定例を以下に示す。

例 1

オリジナルの文字列 U+05E9,U+05BC,U+05C1,U+05E9,U+05C2,U+05E9,U+05C1 に対して、ligature の処理などで、

U+05E9,U+05BC,U+05C1 が U+FB2C

U+05E9,U+05C2 が U+FB2B

Ul+05E9,U+05C1 が U+FB2A

となるような場合、lpUStr1、lpUStr2、UstrMap には以下のように設定する。

```
lpUStr1[0]=0x05E9; lpUStr1[1]=0x05BC; lpUStr1[2]=0x05C1;
lpUStr1[3]=0x05E9;lpUStr1[4]=0x05C2;lpUStr1[5]=0x05E9;
lpUStr1[6]=0x05C1;
lpUStr2[0]=0xFB2C;lpUStr2[1]=0xFB2B;lpUStr2[2]=0xFB2A;
UStrMap.StrNum=3;
UStrMap.lpIdx[0].SrcPos=0;UStrMap.lpIdx[0].DstPos=0;
UStrMap.lpIdx[1].SrcPos=3;UStrMap.lpIdx[1].DstPos=1;
UStrMap.lpIdx[2].SrcPos=5;UStrMap.lpIdx[2].DstPos=2;
```

例 2

例 2 の逆に、オリジナルの文字列が U+FB2C,U+FB2B,U+FB2A 、この U+FB2C が U+05E9,U+05BC、U+FB2B が U+05C1,U+05E9、U+FB2A が U+05C2、U+05E9 となるよう

な場合は以下のように設定する。

lpUStr1[0]=0xFB2C; lpUStr1[1]=0xFB2B; lpUStr1[2]=0xFB2A

lpUStr2[0]=0x05E9; lpUStr2[1]=0x05BC; lpUStr2[2]=0x05C1;

lpUStr2[3]=0x05E9; lpUStr2[4]=0x05C2; lpUStr2[5]=0x05E9;

UStrMap.StrNum=3;

UStrMap.lpIdx[0].SrcPos=0; UStrMap.lpIdx[0].DstPos=0;

UStrMap.lpIdx[1].SrcPos=1; UStrMap.lpIdx[1].DstPos=2;

UStrMap.lpIdx[2].SrcPos=2; UStrMap.lpIdx[2].DstPos=4;

例 3

元の文字列が U+05E9,U+05BC,U+05C1,U+05E9,U+05C2,U+05E9,U+05C1 の先頭 4 文字が U+FB2C,U+FB2B に、残りの 3 文字が、U+FB2A に対応する場合、以下のように設定する。

lpUStr1[0]=0x05E9; lpUStr1[1]=0x05BC; lpUStr1[2]=0x05C1;

lpUStr1[3]=0x05E9;lpUStr1[4]=0x05C2; lpUStr1[5]=0x05E9;

lpUStr1[6]=0x05C1;

lpUStr2[0]=0xFB2C; lpUStr2[1]=0xFB2B; lpUStr2[2]=0xFB2A

UStrMap.StrNum=2;

UStrMap.lpIdx[0].SrcPos=0; UStrMap.lpIdx[0].DstPos=0;

UStrMap.lpIdx[1].SrcPos=4;　UStrMap.lpIdx[1].DstPos=2;

8.6.9. グリフ番号指定による文字列出力関数

PDFAPI PL_ERROR pl_ShowTextGI(hPDF ctlp,const UCHAR_t* lpUStr,const PL_GStrTD& GlyphStr, const PL_StrMapTD& UGMap)

機能：

カレント位置に指定されたグリフ番号の文字列を出力する。

引数：

ctlp：PDF ファイルのポインタ

lpUStr：システム Endian と一致する Unicode で出力するグリフ番号の元の文字列を指定する。

GlyphStr：グリフ番号、及びその幅を示す。

UGMap：lpUStr と GlyphStr のそのグリフの対応付けを示す。PL_StrMapTD の定義は pl_ShowTextU2 の説明を参照のこと。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明:

この関数の pl_ShowTextU()関数及び pl_ShowTextU2()関数との違いは、出力する文字列をグリフ番号で指定することにある。主にヘブライ語、アラビア語など、OpenType/TrueType フォントの Glyph Substitution などを呼び出し側で処理済みの場合を想定したものである。

TrueType,OpenType 以外のフォントで使用してはならない。

PL_GStrTD の定義を以下に示す。

```
typedef struct {
    int GlyphNum;            // lpGInfo の長さ。PL_StrMapTD の StrNum と同じであること。
    PL_GlyphInfoTD* lpGInfo;  // グリフの番号および幅
} PL_GStrTD;
```

PL_GlyphInfoTD の定義を以下に示す。

```
typedef struct {
    GlyphID      Glyph;          // グリフ番号
    int Glyph    Width;          // グリフ幅
} PL_GlyphInfoTD;
```

GlyphWidth には PDFcreator.h に定義される PL_GW_DEFAULTUNIT を単位とした値を指定する。

なお、lpUStr と GlyphStr の各要素の対応は UGMap で指定する。この関係は pl_ShowTextU2 の lpUStr1、lpUStr2、UStrMap の対応と同じである。

埋め込みフォント設定関数 pl_PutEmbOpt( ) の lpSetEmbOpt.UnusualAction に、PL_EMBEDTRUE が指定されている場合、ここで出力されたフォントは埋め込みが行われる。PL_EMBEDFALSE が指定され場合、Identity-H エンコーディングを使用した、非埋め込みの出力となる。

8.6.10. 文字幅計算関数

PDFAPI PL_ERROR pl_GetCharWidth(hPDF ctlp,const UCHAR_t fc,float* pcw)

機能：

文字の幅を計算するための支援関数である。pl_SetFont()で設定されているフォントを使用して計算する。

引数：

ctlp ： PDF ファイルのポインタ

fc ： 文字

pcw ：文字の幅

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.6.11. 文字列表示幅計算関数

PDFAPI PL_ERROR pl_GetStrWidthU(hPDF ctlp, const UCHAR_t* ustrp,float* psw)

機能：

Uicode 文字列の幅を計算するための支援関数である。pl_SetFont()で設定されているフォントを使用して計算する。

引数：

ctlp ： PDF ファイルのポインタ

ustrp ： Unicode 文字列

psw ： Unicode 文字列の幅

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.6.12. 文字列表示幅計算関数 2

PDFAPI PL_ERROR pl_GetStrWidthU2(hPDF ctlp, const UCHAR_t* lpUStr,float* psw)

機能：

Unicode 文字列の幅を計算する。pl_SetFont()で設定されるフォントを使用して計算する。

引数：

ctlp ： PDF ファイルのポインタ

lpUStr： 変換先の Unicode 文字列

psw：Unicode 文字列の幅

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明:

この関数は pl_ShowTextU2()関数と対応する。pl_GetStrWidthU()関数との違いは、この関数の lpUStr の対象となる文字列は pl_ShowTextU2()の lpUStr2 で使用する変換後の Unicode 文字列であることにある。

8.6.13. ミッシンググリフ文字コード取得関数

PDFAPI void pl_GetMissGlyphChar (hPDF ctlp , UCHAR_t* lpMissChar)

機能：

pl_ShowTextU 関数を使用して文字を出力する場合、フォントが対応するグリフを持たない場合、pl_PutMissGlyphOpt の指定によっては、PL_ERR_F_QueryGlyExist が戻る。このときに、ミッシンググリフとなった文字のコードを取得することができる。

引数：

ctlp： PDF ファイルポインタ

lpMissChar：グリフが存在しない文字のコード

戻り値：

なし

8.6.14. ミッシンググリフフォント名取得関数

PDFAPI void pl_GetCurAssignedFont(hPDF ctlp, UCHAR_t* lpCurFont)

機能：

ミッシンググリフが発生した場合に、そのフォントの名称を取得することができる。

引数：

ctlp ： PDF ファイルポインタ

lpCurFont：グリフが存在しない文字のフォント名

戻り値：

なし

8.7. イメージ操作関数

イメージデータを出力する場合に使用する関数群である。

- ファイルインタフェース方式の Image 出力

通常のイメージデータの場合の呼出しシーケンス

pl_CheckImage() ⇒ pl_InitImage() ⇒ pl_OpenImage() ⇒ pl_PutImage() ⇒ pl_CloseImage() ⇒ pl_FinishImage()。このうち、pl_CheckImage()は省略可能であるが、ほかの関数は必ずこの順で使用すること。

マスクを指定してイメージを表示する場合の呼出し順

pl_CheckImage() ⇒ pl_InitImage() ⇒ pl_CheckImageMask() ⇒ pl_InitImageMask() ⇒ pl_OpenImageWithMask() ⇒ pl_PutImage() ⇒ pl_CloseImage() ⇒ pl_FinishImage()。このうち、pl_CheckImage()と pl_CheckImageMask()は省略可能であるが、ほかの関数は必ず省略できない。組合せて使用すること。

- Stream インタフェース方式の Image 出力

  イメージのストリームインタフェースが二つのセットである。一つはファイルインタフェースと同様の呼び出し順による処理であり、もう一つは複数回同一画像を出力する場合などに適したインタフェースである。

  - 第一セットのストリームインタフェースに下記の関数が含まれる
    - pl_InitImage()
    - pl_InitImageMask()
    - pl_CheckImage_Stream()
    - pl_CheckImageMask_Stream()
    - pl_OpenImage_Stream()
    - pl_OpenImageWithMask_Stream()
    - pl_CheckColorProfile_Stream()
    - pl_GetImageColorProfile_Stream()
    - pl_PutImage()
    - pl_CloseImage()

✧ pl_FinishImage()

通常のイメージデータの場合の呼出しシーケンス

pl_CheckImage_Stream() ⇒ pl_InitImage_Stream() ⇒ pl_OpenImage_Stream() ⇒ pl_PutImage()⇒pl_CloseImage()⇒pl_FinishImage()。このうち、pl_CheckImage()は省略可能であるが、ほかの関数は必ずこの順で使用すること。

マスクを指定してイメージを表示する場合の呼出し順

pl_CheckImage_Stream() ⇒ pl_InitImage_Stream() ⇒ pl_CheckImageMask_Stream() ⇒ pl_InitImageMask_Stream() ⇒ pl_OpenImageWithMask_Stream() ⇒ pl_PutImage() ⇒ pl_CloseImage()⇒pl_FinishImage()。このうち、pl_CheckImage()と pl_CheckImageMask()は省略可能であるが、ほかの関数は必ず省略できない。組合せて使用すること。

◆ 第ニセットのストリームに下記の関数が含まれる

✧ pl_CheckImage_Stream()

✧ pl_CheckImageMask_Stream()

✧ pl_LoadImage_Stream()

✧ pl_LoadImageMask_Stream()

✧ pl_LoadImageWithMask_Stream()

✧ pl_PutImageByHandle()

✧ pl_RemoveImageByHandle()

✧ pl_GetImageColorProfileByHandle()

✧ pl_GetImageColorProfileFByHandle_Stream()

✧ pl_GetImageWHByHandle()

✧ pl_ColorizeImageByHandle()

通常のイメージデータの場合の呼出しシーケンス

pl_CheckImage_Stream() ⇒ pl_LoadImage_Stream() ⇒ pl_PutImageByHandle () ⇒ pl_RemoveImageByHandle()。

このうち、pl_CheckImage_Stream ()と pl_RemoveImageByHandle ()は省略可能であるが、ほかの関数は必ずこの順で使用すること。

マスクイメージデータの場合の呼出しシーケンス

pl_CheckImageMask_Stream()⇒pl_LoadImageMask_Stream()⇒pl_PutImageByHandle ()⇒ pl_RemoveImageByHandle()。

このうち、pl_CheckImageMask_Stream ()和 pl_RemoveImageByHandle ()は省略可能であるが、ほかの関数は必ずこの順で使用すること。

マスクを指定してイメージを表示する場合の呼出し順

pl_CheckImage_Stream() ⇒ pl_CheckImageMask_Stream() ⇒

pl_LoadImageWithMask_Stream() ⇒ pl_PutImageByHandle() ⇒ pl_RemoveImageByHandle()。

このうち、pl_CheckImage_Stream (),pl_CheckImageMask_Stream(), pl_RemoveImageByHandle() は省略可能であるが、ほかの関数は必ずこの順で使用すること。

8.7.1. イメージテスト関数

PDFAPI PL_ERROR pl_CheckImage(hPDF ctlp, const char* fn, int frameNo = 0)

機能：

本ライブラリで出力可能なイメージ形式であるか否かを判定し、その結果を戻す。

引数：

ctlp：PDF ファイルのポインタ

fn：イメージファイルの名称

frameNo：マルチフレーム構造の TIFF に対して、フレーム番号を指定する

戻り値：

サポートされる画像の場合、0 を戻し、それ以外の場合、エラーコードを戻す

説明：

サポート可能なイメージ形式であった場合、以降の関数でイメージを処理することができる。サポートされないイメージ形式の場合、以降の関数を呼び出してはならない。

サポート可能な画像には、PDF にパススルーで格納可能な形式、あるいは解凍などの何らかの加工を行った後、格納可能となる形式の双方が含まれる。

PDF/X ではアルファチャンネル付きの画像はサポートされない。

PDF/X-1 では、DeviceRGB、CalRGB、Lab、ICCBased のカラースペースを持つ画像をサポートしない。また、LZW 圧縮方式を使用した画像をサポートしない。

PDF/X-3 に対して、画像の ColorSpace に制限がないが、以下の規則に準ずる必要がある。

カラースペースが RGB である画像を出力する場合、出力デバイスが RGB カラースペースをサポートしない場合、DefaultRGB を設定しなければならない。GRAY、CMYK も同様である。

8.7.2. イメージマスク検査関数

PDFAPI PL_ERROR pl_CheckImageMask(hPDF ctlp, const char* fn, int frameNo = 0)

機能 ：

ライブラリがサポートしているマスクイメージの形式であるか否かを判定し、その結果を戻す。マスクに使用するためには単色の画像かつ、BitsPerComponent が 1 でなければならない。

引数 ：

ctlp ： PDF ファイルのポインタ

fn ： イメージファイルの名称

frameNo：マルチフレーム構造の TIFF に対して、フレーム番号を指定する

戻り値 ：

サポートされる画像の場合、0 を戻し、それ以外の場合、エラーコードを戻す

説明:

サポート可能なイメージ形式であった場合、以降の関数でこれをマスクイメージとして処理することができる。サポートされないイメージ形式の場合、以降の関数を呼び出してはならない。

8.7.3. イメージ初期化関数

PDFAPI PL_ERROR pl_InitImage(hPDF ctlp, hIMG* pImgp)

機能：

イメージ処理で使用するメモリの獲得などを行う。

引数：

ctlp：PDF ファイルのポインタ

pImgp：イメージデータのポインタのアドレス

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

イメージを処理する前に、この関数を呼ぶ必要がある。

8.7.4. イメージマスク初期化関数

PDFAPI PL_ERROR pl_InitImageMask(hPDF ctlp, hIMG* pImgp,
PL_MaskValueE maskValue=PL_MV_NOTZERO)

機能 ：

イメージマスク処理で使用するメモリの獲得などを行う。

引数 ：

ctlp：PDF ファイルのポインタ

pImgp：イメージデータのポインタアドレス

maskValue：マスクの Decode 配列の設定を指定する。定義を以下に示す。

```
typedef enum{
        PL_MV_NOTZERO    = 0,     //  1 をマスク(Decode に[0 1])
        PL_MV_ZERO       = 1,     //  0 をマスク(Decode に[1 0])
        PL_MV_WHITE      = 2,     //  ホワイトポイントをマスク
        PL_MV_NOTWHITE   = 3      //  ホワイトポイントを非マスク
} PL_MaskValueE;戻り値：
```

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

イメージマスクを処理する前、この関数を呼ぶ必要がある。

8.7.5. イメージオープン関数

PDFAPI PL_ERROR pl_OpenImage(hPDF ctlp, const char* fn, hIMG pImg)

機能：

イメージファイルをオープンする。

引数：

ctlp：PDF ファイルのポインタ

fn：イメージファイル名

pImg：イメージデータのポインタ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.7.6. イメージオープン関数(ストリーム)

PDFAPI PL_ERROR pl_OpenImage_Stream(hPDF ctlp,std::istream& ismImage,hIMG pImg)

機能：

イメージファイルをオープンする。

引数：

ctlp：PDF ファイルのポインタ

ismImage：イメージストリーム

pImg：イメージデータのポインタ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.7.7. マスク付きイメージのオープン関数

PDFAPI PL_ERROR pl_OpenImageWithMask(hPDF ctlp,const char* ifn,const char* mfn, hIMG pImg)

機能：

マスク付きイメージファイルをオープンする。マスクイメージは色成分が一つであり、さらに、BitsPerComponent が 1 である必要がある。

引数：

ctlp ： PDF ファイルのポインタ

ifn ： イメージファイル名

mfn ： マスクイメージファイル名

pImg ： イメージデータのポインタ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.7.8. マスク付きイメージのオープン関数(ストリーム)

PDFAPI PL_ERROR pl_OpenImageWithMask_Stream(hPDF ctlp,std::istream& ismImage, std::istream& ismImageMask,hIMG pImg)

機能：

マスク付きイメージファイルをオープンする。マスクイメージは色成分が一つであり、さらに、BitsPerComponent が 1 である必要がある。

引数：

ctlp ： PDF ファイルのポインタ

ismImage ： イメージ Stream

ismImageMask ： マスクイメージ Stream

pImg ： イメージデータのポインタ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.7.9. イメージ出力関数

PDFAPI PL_ERROR pl_PutImage(hPDF ctlp, hIMG pImg,float x,float y, float w,float h)

機能：

指定された位置にイメージを出力する。

引数：

ctlp： PDF ファイルのポインタ

pImg：イメージデータのポインタ

x,y ：イメージの左下隅のカレントページ上での位置を指定する

w,h：イメージのカレントページ上での幅(w)と高さ(h)を指定する

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.7.10. イメージクローズ関数

PDFAPI PL_ERROR pl_CloseImage(hPDF ctlp, hIMG pImg)

機能：

イメージファイルを閉じて、一部のメモリを開放する。

引数：

ctlp：PDF ファイルのポインタ

pImgp：イメージデータのポインタ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.7.11. イメージ終了関数

PDFAPI void pl_FinishImage(hPDF ctlp, hIMG* pImgp)

機能：

イメージデータのポインタのメモリを開放する。

引数：

ctlp：PDF ファイルのポインタ

pImgp：イメージデータのポインタのアドレス

戻り値：

なし

8.7.12. グレースケールイメージ着色関数

PDFAPI void pl_ColorizeImage(hPDF ctlp, hIMG pImg,int spotCSIdx)

機能：

グレースケールイメージデータをセパレーションカラースペースで着色して表示させる。

引数：

ctlp：PDF ファイルのポインタ

pImg：イメージデータハンドル

spotCSIdx：カラースペースのインデクス

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

PL_ERR_IMG_UnColorized ：グレースケールでないイメージが指定された、あるいは、セパレーションでないカラースペースが spotCSIdx で指定された。

説明：

グレースケールイメージ(カラー成分が 1 種類)のデータのカラースペースを、スポットカラーとすることで、指定のインクによる着色表示を行う機能を提供する。

8.7.13. イメージサイズ取得関数

PDFAPI PL_ERROR pl_GetImageWH(hPDF ctlp,const char* fn,float& iWidth,float& iHeight, int frameNo = 0)

機能：

指定されるイメージの幅と高さを取得する。

引数：

ctlp ： PDF ファイルのポインタ

fn ： イメージファイル名

iWidth ： 取得するイメージの幅

iHeigh ： 取得するイメージの高さ

frameNo：マルチフレーム構造の TIFF に対して、フレーム番号を指定する

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.7.14. イメージサイズ取得関数(ストリーム)

PDFAPI PL_ERROR pl_GetImageWH_Stream(hPDF ctlp,std::istream& ismImage,float& iWidth,float& iHeight,int frameNo = 0)

機能：

指定されるイメージの幅と高さを取得する。

引数：

ctlp ： PDF ファイルのポインタ

ismImage ： イメージ Stream

iWidth ： 取得するイメージの幅

iHeigh ： 取得するイメージの高さ

frameNo：マルチフレーム構造の TIFF に対して、フレーム番号を指定する

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.7.15. イメージカラープロファイル取得関数

PDFAPI PL_CPHandle pl_GetImageColorProfile(hPDF ctlp, hIMG pImg)

機能：

イメージデータ内のカラープロファイルを取得する。

引数：

ctlp ： PDF ファイルポインタ

pImg：イメージハンドル（pl_OpenImage で取得する）

cpInfoType：取得する情報を指定する

戻り値：

イメージデータ内のカラープロファイルのハンドルを戻す。取得できない場合、0 を戻す。

8.7.16. イメージカラープロファイル取得関数(ファイル作成)

PDFAPI PL_ERROR pl_GetImageColorProfileF(hPDF ctlp, hIMG pImg,const char * outColorProfile)

機能：

イメージデータ内のカラープロファイルをファイルとして取り出す。

引数：

ctlp ： PDF ファイルポインタ

pImg：イメージハンドル（pl_OpenImage で取得する）

outColorProfile：作成するカラープロファイルのファイル名を指定する

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.7.17. イメージカラープロファイル取得関数(ストリーム作成)

PDFAPI PL_ERROR pl_GetImageColorProfile_Stream(hPDF ctlp,hIMG pImg,std::ostream& osmOutColorProfile)

機能：

イメージデータ内のカラープロファイルをファイルとして取り出す。

引数：

ctlp ： PDF ファイルポインタ

pImg：イメージハンドル（pl_OpenImage で取得する）

osmOutColorProfile：作成するカラープロファイルの stream を指定する

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.7.18. イメージ検査関数(Stream 版)

PDFAPI PL_ERROR pl_CheckImage_Stream(hPDF ctlp,std::istream& ismImage);

機能：

本ライブラリで出力可能なイメージ形式であるか否かを判定し、その結果を戻す。

引数：

ctlp：PDF ファイルのポインタ

ismImage：イメージ stream.

戻り値：

サポートされる画像の場合、0 を戻し、それ以外の場合、エラーコードを戻す

説明：

pl_CheckImage を参照

8.7.19. イメージマスク検査関数(Stream 版)

PDFAPI PL_ERROR pl_CheckImageMask_Stream(hPDF ctlp,std::istream& ismImage);

機能 ：

ライブラリがサポートしているマスクイメージの形式であるか否かを判定し、その結果を戻す。マスクに使用するためには単色の画像かつ、BitsPerComponent が 1 でなければならない。

引数 ：

ctlp ： PDF ファイルのポインタ

ismImage ： イメージマスクの stream

戻り値 ：

サポートされる画像の場合、0 を戻し、それ以外の場合、エラーコードを戻す

説明:

pl_CheckImageMask を参照

8.7.20. イメージのロード関数(Stream 版)

PDFAPI PL_ERROR pl_LoadImage_Stream(hPDF ctlp,std::istream& ismImage,PL_ImgHandle& imgHandle);

機能：

イメージをロードする。

引数：

ctlp：PDF ファイルのポインタ

ismImage：イメージの Stream

imgHandle：イメージデータのハンドル

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.7.21. マスクイメージのロード関数(Stream 版)

PDFAPI PL_ERROR pl_LoadImageMask_Stream(hPDF ctlp,std::istream& ismImage,PL_ImgHandle& imgHandle);

機能：

マスクイメージをロードする。

引数：

ctlp ： PDF ファイルのポインタ

ismImage：マスクイメージの Stream

imgHandle：マスクイメージデータのハンドル

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.7.22. マスク付きイメージのロード関数(Stream 版)

PDFAPI PL_ERROR pl_LoadImageWithMask_Stream(hPDF ctlp,std::istream& ismImage,
std::istream& ismImageMask,PL_ImgHandle& imgHandle);

機能：

マスク付きイメージをロードする。

引数：

ctlp：PDF ファイルのポインタ

ismImage：イメージの Stream

ismImageMask：マスクイメージの Stream

imgHandle：イメージデータのハンドル

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.7.23. イメージ出力関数(Image Handle 指定)

PDFAPI PL_ERROR pl_PutImageByHandle (hPDF ctlp,PL_ImgHandle imgHandle,float x,float y, float w,float h);

機能：

指定された位置にイメージを出力する。

引数：

ctlp： PDF ファイルのポインタ

imgHandle：イメージデータのハンドル

x,y ：イメージの左下隅のカレントページ上での位置を指定する

w,h：イメージのカレントページ上での幅(w)と高さ(h)を指定する

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.7.24. イメージの削除関数(Stream 版)

PDFAPI PL_ERROR pl_RemoveImageByHandle (hPDF ctlp,PL_ImgHandle imgHandle);

機能：

メモリから imgHandle に指される画像データを削除する。

Load 関数により取得した imgHandle を使用して pl_PutImageByHandle()で出力する。このとき、PDF ファイル内に同じ画像を出力せず、単一の画像を出力し、これを共用するように設定する。

pl_PutImage,および pl_PutImage()も、同様に画像データの共用設定を行うが、画像データの比較処理が行われるため、pl_PutImageByHandle()の使用を推奨する。

この関数を呼び出さない場合、データの削除は pl_ClosePdf 関数の呼び出し時点で行われる。

引数：

ctlp ： PDF ファイルのポインタ

ismImage：マスクイメージの Stream

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.7.25. イメージカラープロファイル取得関数(Image Handle 指定)

PDFAPI PL_ERROR pl_GetImageColorProfileByHandle(hPDF ctlp,PL_ImgHandle imgHandle,PL_CPHandle& hProfile);

機能：

イメージデータ内のカラープロファイルを取得する。

引数：

ctlp ： PDF ファイルポインタ

imgHandle：イメージハンドル（pl_LoadImage で取得する）

h Profile: イメージデータ内のカラープロファイルのハンドル

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.7.26. イメージカラープロファイル取得関数(Stream 作成, Image Handle 指定)

```
PDFAPI PL_ERROR pl_GetImageColorProfileFByHandle_Stream(hPDF ctlp, PL_ImgHandle imgHandle,std::ostream& osmOutColorProfile);
```

機能：

イメージデータ内のカラープロファイルを Stream として取り出す。

引数：

ctlp ： PDF ファイルポインタ

imgHandle：イメージハンドル（pl_LoadImage で取得する）

osmOutColorProfile：作成するカラープロファイルの stream

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.7.27. イメージサイズ取得関数(Image Handle 指定)

```
PDFAPI PL_ERROR pl_GetImageWHByHandle(hPDF ctlp,PL_ImgHandle imgHandle,
                                      float& iWidth,float& iHeight);
```

機能：

指定されるイメージの幅と高さを取得する。

引数：

ctlp ： PDF ファイルのポインタ

imgHandle：イメージハンドル（pl_LoadImage で取得する）

iWidth ： 取得するイメージの幅

iHeigh ： 取得するイメージの高さ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.7.28. グレースケールイメージ着色関数(イメージハンドル指定)

```
PDFAPI PL_ERROR pl_ColorizeImageByHandle(hPDF ctlp,PL_ImgHandle imgHandle,int spotCSIdx)
```

機能：

グレースケールイメージデータをセパレーションカラースペースで着色して表示させる。

引数：

ctlp：PDF ファイルのポインタ

imgHandle：イメージハンドル（pl_LoadImage で取得する）

spotCSIdx：カラースペースのインデクス

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

PL_ERR_IMG_UnColorized ：グレースケールでないイメージが指定された、あるいは、セパレーションでないカラースペースが spotCSIdx で指定された。

説明：

グレースケールイメージ(カラー成分が 1 種類)のデータのカラースペースを、スポットカラーとすることで、指定のインクによる着色表示を行う機能を提供する。

8.7.29. フレーム数取得関数

PDFAPI PL_ERROR pl_GetImageFrameNum(hPDF ctlp,hIMG pImg,int & imgFrameNum, int & imgMaskFrameNum)

機能：

TIFF のマルチフレーム画像のフレーム数を取得する。

引数：

ctlp：PDF ファイルのポインタ

pImg：イメージデータのポインタ

imgFrameNum：フレーム数

imgMaskFrameNum：マスクフレーム数

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.7.30. フレーム数取得関数(ハンドル指定)

PDFAPI PL_ERROR pl_GetImageFrameNumByHandle(hPDF ctlp,PL_ImgHandle imgHandle,int & imgFrameNum, int & imgMaskFrameNum)

機能：

TIFF のマルチフレーム画像のフレーム数を取得する。

引数：

ctlp：PDF ファイルのポインタ

imgHandle：イメージハンドル (pl_LoadImage で取得する)

imgFrameNum：フレーム数

imgMaskFrameNum：マスクフレーム数

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.7.31. フレーム番号設定関数

PDFAPI PL_ERROR pl_SetImageFrameNo(hPDF ctlp,hIMG pImg,int imgFrameNum, int imgMaskFrameNum = 0)

機能：

TIFF のマルチフレーム画像のフレーム番号を設定する。

引数：

ctlp：PDF ファイルのポインタ

pImg：イメージデータのポインタ

imgFrameNum：フレーム番号

imgMaskFrameNum：マスクフレーム番号

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.7.32. フレーム番号設定関数(ハンドル指定)

PDFAPI PL_ERROR pl_SetImageFrameNoByHandle(hPDF ctlp,PL_ImgHandle imgHandle,int imgFrameNum, int imgMaskFrameNum = 0)

機能：

TIFF のマルチフレーム画像のフレーム番号を設定する。

引数：

ctlp：PDF ファイルのポインタ

imgHandle：イメージハンドル（pl_LoadImage で取得する）

imgFrameNum：フレーム番号

imgMaskFrameNum：マスクフレーム番号

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.7.33. カラーキー設定関数

PDFAPI PL_ERROR pl_SetImageColorKey(hPDF ctlp,hIMG pImg,unsigned int startKey,unsigned int endKey)

機能：

カラーキーを設定する。

引数：

ctlp：PDF ファイルのポインタ

pImg：イメージデータのポインタ

startKey：開始キー番号

endKey：終了キー番号

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.7.34. カラーキー設定(ハンドル指定)

PDFAPI PL_ERROR pl_SetImageColorKeyByHandle(hPDF ctlp,PL_ImgHandle imgHandle,unsigned int startKey,unsigned int endKey)

機能：

カラーキーを設定する。

引数：

ctlp：PDF ファイルのポインタ

imgHandle：イメージハンドル（pl_LoadImage で取得する）

startKey：開始キー番号

endKey：終了キー番号

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.8. 関数オブジェクト出力関数

PDF ファイルでは関数オブジェクトがサポートされる。このオブジェクトを使用する場合、以下の関数を使用する。本ライブラリでは、関数オブジェクトはシェーディングオブジェクトおよび Separation カラースペースの色調変換関数の指定で使用する。関数オブジェクトについての説明は PDF 仕様書の「3.9 関数」参照。

呼び出すシーケンス：以下の順で使用すること。

pl_InitFunction()⇒pl_CreateFunction()⇒pl_FreeFunction()

8.8.1. 関数オブジェクト初期化関数

PDFAPI PL_ERROR pl_InitFunction(hPDF ctlp, PL_FunTypeE type ,int m,int n,PL_FunctionTD pfctp)**

機能：

関数オブジェクトを初期化する

引数：

ctlp： PDF ファイルのポインタ

type：関数オブジェクトの型を指定する。

（PDF1.3～1.7 では 0、2、3、4 の四種類の型がサポートされている）

m：関数の入力引数の個数(Domain 配列は 2×m 個の配列となる)

n：関数の出力引数の個数(Range 配列は 2×n 個の配列となる)

pfctp：関数オブジェクトのポインタのアドレス。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_FunTypeE の定義を以下に示す。

```
typedef enum{
    PL_FUNCTION_SAMPLE=0,              // タイプ 0(サンプリング関数)
    PL_FUNCTION_EXPONENTIAL=2,         // タイプ 2 (指数補間関数)
    PL_FUNCTION_STITCHING,             // タイプ 3 (縫合関数)
    PL_FUNCTION_CALCULATOR             // タイプ 4 (PostScript 計算関数)
} PL_FunTypeE;
```

Type3 タイプの関数では、m、n は必ず一致する必要があり、これは関数の個数を示す。各関数は、全て 1 入力、1 出力関数であり、Domain のパラメータ個数は 2 である。

PL_FunctionTD の定義を以下に示す。

```
typedef struct {
    /*  全関数タイプ共通  */
    PL_FunTypeE  FunctionType;     /*関数タイプ(0,2,3,4)*/
    int          m,n;              /* m：入力値の数、n：出力値の数
                                   ただし、タイプ 3 関数の場合、PDF 仕様書の k の値を  m  に
                                   設定する*/
    float        *Domain;          /* Domain 配列の要素(2  ×  m  個)、
                                   ただし、タイプ 3 関数では、2 個固定*/
    float        *Range;           /* Range 配列の要素(2  ×  n  個)
                                    ただし、タイプ 3 関数では 2 個固定*/
    PL_FuncOptParaE    setFlag     /*  パラメータの設定状態を定義する  */
    /*  タイプ 0  関数*/
    int          *Size;            /*Size 配列の要素(m 個) */
    int          BitPerSample;     /*サンプルのビット数(1,2,4,8,12,16,24,32)*/
    int          Order;            /*補間方式(1：線形、3：スプライン)*/
    float        *Encode;          /* Encode 配列の要素(2  ×  m 個)*/
    float        *Decode;          /* Decode 配列の要素(2  ×  n 個)*/
    /*  タイプ 2 関数  */
    float        *C0;              /* C0 配列の要素(n 個) */
    float        *C1;              /* C1 配列の要素(n 個) */
    float        N;                /*  補間指数  */
    /*  タイプ 3 関数  */
    int*         funcIdxp          /*  関数インデクス*/
    float        *Bounds;          /* Bounds 配列(n-1 個)
                                   Encode 配列はタイプ 0 関数と共用*/
    /*  タイプ 0 関数とタイプ 4 関数  */
    PL_MemStreamTD stream;         /*ストリーム
                                   ユーザ側でメモリを確保して内容を設定する*/
```

} PLFunctionTD;

各パラメータを設定した場合、setFlag に下記の対応するフラグをセットする必要がある。

```
typedef enum {
        PL_FOP_ORDER   = 1 << 0,        //for type0
        PL_FOP_ENCODE = 1 << 1,         //for type0
        PL_FOP_DECODE = 1 << 2,         //for type0
        PL_FOP_C0      = 1 << 3,        //for type2
        PL_FOP_C1      = 1 << 4,        //for type2
        PL_FOP_RANGE   = 1 << 5         //for type2 and type3
} PL_FuncOptParaE;
```

PL_MemStreamTD の定義を以下に示す。

```
typedef struct
{
    int    Length;          // データ長(byte 数)
    char   *data;           // データ
} PL_MemStreamTD;
```

8.8.2. 関数オブジェクト作成関数

PDFAPI PL_ERROR pl_CreateFunction(hPDF ctlp,PL_FunctionTD* fctlp, int& funcIdx)

機能：

関数オブジェクトを作成する。

引数：

ctlp：PDF ファイルのポインタ

fctlp：関数オブジェクトのポインタ。pl_InitFunction 関数から戻ったポインタ。

この関数を呼び出す前に、関数ごとのデータを、この構造体に設定する必要がある。なお、構造体メンバー FunctionType、m は pl_InitFunction 関数が設定するため、設定不要である。

また、この関数を使用した場合、pl_FreeFunction を呼び出してこのオブジェクトを解放する必要がある。

タイプ 3 関数を指定する場合の、構造体メンバー funcIdxp には縫合する関数を初期化した時に戻される関数のインデクスをセットして呼び出す。

funcIdx：関数インデクスが戻る

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.8.3. 関数オブジェクトフリー関数

PDFAPI PL_ERROR pl_FreeFunction(hPDF ctlp, PL_FunctionTD pfctp)**

機能：

関数オブジェクトを解放する。

引数：

ctlp ： PDF ファイルのポインタ

pfctp ： function オブジェクトのポインタのアドレス。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.9. パターン定義関数

パターンは図形の塗りつぶしの一種である。色と同様に線の描画と図形の塗りつぶしに使われる。PDF には、タイプ 1 (タイリングパターン)、タイプ 2 (シェーディング)の 2 種類のパターンが定義される。また、タイプ 1 には色つき、および色無しの 2 種類が定義される。

シェーディングは、領域内の光、色の変換を表現する。例えば、ボールの表面の光と色の変換によって、ボールの立体感を表現することができる。PDF では、7 種類のシェーディングがサポートされる。下記関数を使用して PDF ファイルにタイリングパターン、シェーディングパターンを指定することができる。

詳細については PDF 仕様書参照。

8.9.1. タイプ 1 パターン作成開始関数

PDFAPI PL_ERROR pl_BgnPattern1(hPDF ctlp, PL_Pattern1TD* patp)

機能：

タイプ 1 のパターンの作成を開始する。

引数：

ctlp：PDF ファイルのポインタ

patp：タイプ 1 パターン構造体へのポインタ。

pattern1 の構造体に、関係する引数を設定する。pattern1 は小さいページであり、この関数を呼び出してから、pl_endpattern1 関数を呼び出すまで、通常のページに対するすべての描画操作は、このパターンに対する動作となり、パターンデータとして登録される。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

Pattern1TD の定義を以下に示す。

```
typedef struct {
    PL_PaintTypeE PaintType;        /* ペイントタイプ */
    PL_TilingTypeE TilingType;      /* タイリングタイプ */
    PL_RectangleTD BBox;            /* パターンセルを囲む境界ボックスの座標
                                       クリップにも使用される*/
    float          XStep;           /*パターン座標系で測定された望ましい
```

```
                                        水平パターンセル間隔*/
    float       YStep;          /*パターン座標系で測定された望ましい
                                        水平パターンセル間隔*/
    HBOOL       bMatrixSetFlag; /*Matrix を使用するか否か
                                        HTRUE：使用する。HFALSE：使用しない*/
    float       Matrix[6];      /*変換行列*/
    PatOtherTD  *Otherp;        /*ライブラリ内部使用(設定不要)*/
} PL_Pattern1TD;
```

PL_PaintTypeE の定義を以下に示す。

```
typedef enum{
    PL_PT_COLOREDTILING=1,    // 色付きパターン
    PL_PT_UNCOLOREDTILING     // 色無しパターン
} PL_PaintTypeE;
```

PL_TilingTypeE の定義を以下に示す。

```
typedef enum{
    PL_TT_CONSTSPACE=1,             // 一定間隔
    PL_TT_NODISTOR,                 // 調整無し
    PL_TT_CONSTSPACEFASTER          // 一定間隔で高速タイリング
} PL_TilingTypeE;
```

8.9.2. タイプ１パターン作成終了関数

PDFAPI PL_ERROR pl_EndPattern1(hPDF ctlp,PL_Pattern1TD* patp,int & patIdx)

機能：

タイプ 1 パターンの作成を完了する。パターン作成完了時、本関数を呼び出し、パターンインデクスを取得する。pl_BgnPattern1 関数と組み合わせて使用する必要がある。

引数：

ctlp：PDF ファイルのポインタ

patp：Pattern1TD 構造ポインタ

patIdx：タイプ 1 パターンインデクスを戻す

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.9.3. シェーディングオブジェクト作成関数

PDFAPI PL_ERROR pl_CreateShading(hPDF ctlp,PL_ShadingTD *sdp, int &shIdx)

機能：

シェーディングブジェクトを作成し、シェーディングインデクスを取得する。

引数：

ctlp：PDF ファイルのポインタ

sdp：シェーディングデータオブジェクト。

使用する場合、このオブジェクトのためのメモリを確保・開放を行う必要がある。呼び出し前に、必要なデータを設定のこと。

ShIdx：シェーディングインデクスが戻る

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_ShadingTD の定義を以下に示す。

```
typedef struct {
    PL_ShadTypeE ShadingType;     /*設定値は後述*/
    PL_ColorTypeE sCSType         /* カラースペース*/
    int           sCSIdx          /* カラースペースインデクス
                                        (0 の場合、sCSType 参照)*/
    float         Backgroud[4];   /*背景のカラー*/
    PL_RectangleTD BBox;          /*シェーディングが行われる座標空間における境界ボックスの座標 */
    int           AntiAlias;      /*人工的な「エイリアシング」効果を防ぐためにシェーディング関数をフィルタリングするか否かを指定する */
    // シェーディングタイプ 1、2、3 用追加データ
    int           DomainNum;      /* Domain 配列の要素数*/
    float         Domain[4];      /*シェーディング種別によって意味が異なる*/
    // シェーディングタイプ 1 用追加データ
    HBOOL         bMatrixSetFlag; /* Matrix を使用するか否か
                                     1：使用する 0：使用しない*/
    float         Matrix[6];      /*変換行列(タイプ 1 のみ使用) */
    // シェーディングタイプ 2、3 用追加データ
    int           CoordsNum;      /*Coords の要素数 */
    float         Coords[6];      /*タイプ 2,タイプ 3 のみ使用。タイプによって意味が異なる*/
    int           Extend[2];      /*タイプ 2,タイプ 3 のみ使用。タイプによって意味が異なる*/
    /* シェーディングタイプ 4 用追加データ*/
    int           BitPerFlag;     /*各頂点のエッジフラグを表現するために必要なビット数*/
    int           VerticesPerRow; /*行毎の端点数。2 以上である*/
    /* シェーディングタイプ 4、5、6、7 用追加データ */
    int           BitsPerCoordinate;/*各頂点の座標を表現するために使用されるビット数*/
```

```
    int             BitsPerComponent;/*各カラー成分を表現するために必要なビット数*/
    int             dcnum;          /* Decode の要素数*/
    float           *Decode;        /*  タイプにより意味が異なる*/
    PL_MemStreamTD stream;          //ユーザがメモリ要求を行って内容を設定する必要がある
    /*  以下、全タイプ共通  */
    int             fnum;           /* function の数*/
    int*            funcIdx         /*  関数インデスクの配列をポイントする  */
} PL_ShadingTD;
```

PL_ShadTypeE の定義を以下に示す。

```
typedef enum{
    PL_ST_FUNBASE = 1, PL_ ST_AXIAL, PL_ ST_RADIAL, PL_ ST_FREEFORM,
    PL_ ST_LATTICEFORM, PL_ ST_COONS,PL_ ST_TENSOR
} PL_ShadTypeE;
```

値の意味は以下の通り：

- PL_ ST_FUNBASE : タイプ１「関数ベースのシェーディング」。数学関数を用いて変域に属する各点の色を定義する。
- PL_ ST_AXIAL : タイプ２「軸」シェーディング。2 点を結ぶ線に沿って色のブレンドを定義する。これはオプションで、境界色を続けることによって、境界線を超えて延長される。
- PL_ ST_RADIAL : タイプ３「円形シェーディング」。２つの円の間でブレンドを定義する。これはオプションで、境界色を続けることによって、境界の円を越えて延長される。
- PL_ ST_FREEFORM : タイプ４「フリーフォームのグーローシェーディング三角形メッシュ」。多くの 3 次元アプリケーションが色や影をつけた複雑な形状を表現するために使う、一般的な構造を定義する。三角形の頂点はフリーフォーム幾何形状で指定される。
- PL_ ST_LATTICEFORM : タイプ５「格子状のグーローシェーディング三角形メッシュ」。タイプ４と同じ幾何構造に基づくが、擬似矩形格子として指定される頂点を使用する。
- PL_ ST_COONS : タイプ６「クーンズパッチメッシュ」。1 つ以上のカラーパッチからシェーディングを構成する。個々のカラーパッチは 4 本の 3 次ベジェ曲線を境界とする形状である。
- PL_ ST_TENSOR : タイプ７「テンソル積パッチメッシュ」。各パッチに制御点が追加され、タイプ６より精密なカラーマッピングが可能となっている。

8.9.4. タイプ２パターン作成関数

PDFAPI PL_ERROR pl_CreatePattern2(hPDF ctlp,PL_Pattern2TD* patp,int &patIdx)

機能：

タイプ２のパターンを作成し、パターンインデスクを取得する。

引数：

ctlp：PDF ファイルポインタ

patp：タイプ２パターン構造体へのポインタ

この構造体のメモリを確保し、必要な引数を設定して呼び出す。

patIdx：タイプ２パターンインデクスを戻す

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_ Pattern2TD の定義を以下に示す。

```
typedef struct {
    int         shIdx;          /*シェーディングインデクスを指定する*/
    HBOOL       bMatrixSetFlag; /* Matrix を使用するか否かを指定する
                                   1：使用する。0：使用しない。*/
    float       Matrix[6];      /*変換行列*/
} PL_Pattern2TD;
```

8.9.5. パターン設定関数

PDFAPI PL_ERROR pl_SetPattern(hPDF ctlp,int patIdx, PL_OperatorE mode, PL_ColorTD* color=NULL)

機能：

パターンを塗りつぶしまたはストローク用のカレントカラーとして設定する。あわせて、色無しタイリングパターンの場合はカラー値を設定する。

引数：

ctlp：PDF ファイルのポインタ

patIdx：パターンインデクス(pl_EndPattern1 または pl_CreatePattern2 で取得)

mode：パターンモード（塗りつぶし、ストロークのいずれかを指定する）

PLI_scn： 非ストローク用の色

PLI_SCN： ストロークの色

color：色無しパターンの場合の色を指定する。PL_ColorTD 定義は共通データ形式を参照のこと。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.9.6. シェーディング設定関数

PDFAPI PL_ERROR pl_Shading(hPDF ctlp, int shIdx)

機能：

現在のページにシェーディングを表示する。

引数：

ctlp：PDF ファイルのポインタ

sdp：シェーディングインデクス(pl_CreateShading で取得したもの)

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.10. アウトライン出力関数

アウトライン（しおり）は PDF ファイル中のツリー的な階層構造を持ち、階層構造の各項目はアウトライン項目と呼ばれる。この項目に、本文内の特定の位置などにリンクを指定して、ナビゲーションを行うことができる。PDF ファイルにアウトラインを設定する場合の関数について説明する。

現在の階層の下位に新しい階層を追加して、そこにアウトライン項目を定義すること、および同じ階層内にアウトライン項目を追加することができる。本ライブラリでは、アウトライン項目に設定するリンクの指定方法によって、それぞれ 3 種類の関数をサポートする。

8.10.1. アウトライン階層作成関数

PDFAPI PL_ERROR pl_AddOutlineLevel(hPDF ctlp, PL_DestTD* destp, const UCHAR_t* lpUTitle, HBOOL ExpandFlag)

機能：

新しい階層のアウトラインを作成する。カレントノードに下位の階層を作成する。

引数：

ctlp：ファイルポインタ

destp：作成されたアウトラインが示すターゲット。共通データ形式参照

lpUTitle：表現される outline のタイトルを Unicode で指定する

ExpandFlag：このアウトラインの展開方法を指定するフラグ

HFALSE：閉じた状態、HTRUE：開いた状態

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

このアウトライン項目のリンク先が文書内である場合、あるいは、deststp に NULL を設定してリンク先無しのアウトライン項目とする場合に使用する。

8.10.2. アウトライン階層作成関数 2

PDFAPI PL_ERROR pl_AddOutlineLevel_Local(hPDF ctlp, const PL_OLTitleTD& uTitle, const unsigned char* lpDestName,PL_DestTD* lpDest, HBOOL ExpandFlag);

機能：

新しい階層のアウトラインを作成する。カレントノードに下位の階層が作成される。pl_AddOutlineLevel()関数に対して、名前付き宛先による指定、およびアウトライン項目のスタイル指定機能を追加したものである。リンク先は文書内または無しのいずれかとなる。

引数：

ctlp：ファイルポインタ

uTitle： outline のタイトル、PDF1.4 で追加された色、スタイル属性を指定する。PDF1.3 の PDF

ファイルを作成する場合、tColor と tStyle は無効。OLTitleTD の定義を以下に示す。

```
typedef struct {
    UCHAR_t*        lpUTitle;       // タイトル
    PL_colorTD      tColor;         // タイトルの色(PDF1.4)
    PL_OTStyleE     tStyle;         // タイトルの書式 (PDF1.4)
} PL_OLTitleTD;
```

lpUTitle：アウトラインのタイトル

tColor：アウトライン項目の色であり、RGB で指定する。

tStyle(PDF1.4)：アウトライン項目の書式であり、太字、斜体、太字斜体などを示す。定義を以下に示す

```
typedef enum {
    PL_OT_NONE      = 0 ,
    PL_OT_ITALIC    = 1 << 0 ,      // bit0 が 1 であれば、斜体を示す
    PL_OT_BOLD      = 1 << 1        // bit1 が 1 であれば、太字を示す
} PL_OTStyleE;
```

lpDestName,lpDest：名前または宛先（名前付き宛先の記載参照）.

- いずれか null ではないものが有効である
- 二つともに null ではないなら、lpDestName を使用する
- 二つともに null であるなら、宛先指定の無いしおりを作成する

ExpandFlag：その outline を展開するか否かを示すフラグ。

HFALSE：閉じた状態、HTRUE：開いた状態

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.10.3. リモートアウトライン階層作成関数

```
PDFAPI PL_ERROR pl_AddOutlineLevel_Remote(hPDF ctlp, const PL_OLTitleTD& uTitle,
        PL_FileActionTypeE fActType,  const UCHAR_t* lpFName,const unsigned char* lpDestName,
        PL_DestTD* lpDest,HBOOL ExpandFlag, PL_NewWindowE newWinFlag = PL_NW_NONE);
```

機能：

新しい階層に、ファイル外の宛先をもつアウトライン項目を作成する。カレントノードに下位の階層が作成される。

引数：

ctlp：ファイルポインタ

uTitle,lpDestName,lpDest,ExpandFlag：アウトライン階層作成関数 2 参照

fActType：ファイル外のリンクに対するアクションを指定する。

PL_FileActionTypeE の定義を以下に示す。

```
typedef enum{
```

PL_FA_GOTOR=1, // GoToR アクションでリンクする
PL_FA_LAUNCH, // Launch アクションでリンクする
PL_FA_URI // URI アクションでリンクする
} PL_FileActionTypeE;

lpFName：ファイル名,outline の宛先となる外部ファイルを指定する
null であるなら、宛先指定の無いしおりを作成する.

newWinFlag(optional)：新しいウィンドウで目的ファイルを開くか否かを示す
PL_NewWindow の定義を以下に示す。
typedef enum {
PL_NW_NONE = 0, // newWindow エントリを作成しない
PL_NW_FALSE, // newWindow エントリを作成し、false を設定
// 宛先のファイルは現在の Window 内で開かれる
PL_NW_TRUE // newWindow エントリを作成し、true を設定
// 宛先のファイルは新しい Window 内で開かれる
} PL_NewWindowE;

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.10.4. アウトライン項目作成関数

PDFAPI PL_ERROR pl_AddOutline(hPDF ctlp,PL_DestTD* destp, const UCHAR_t* lpUTitle, HBOOL ExpandFlag)

機能：

現在のアウトラインの階層下にもう一つアウトライン項目を追加する。

引数：

ctlp：PDF ファイルポインタ

destp：作成されるアウトラインが示すデスティネーション

lpUTitle：表示されるアウトラインのタイトルを Unicode で指定する

ExpandFlag：このアウトラインの展開方法を指定するフラグ。
HFALSE：閉じた状態、HTRUE：開いた状態

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.10.5. アウトライン項目作成関数 2

PDFAPI PL_ERROR pl_AddOutline_Local(hPDF ctlp, const PL_OLTitleTD& uTitle, const unsigned char* lpDestName,PL_DestTD* lpDest, HBOOL ExpandFlag);

機能：

現在のアウトラインの階層下にもう一つアウトライン項目を追加する。前の pl_AddOutlineLevel() 関数関数に対して、名前付き宛先による指定、およびアウトライン項目のスタイル指定機能を追加したものである。リンク先は文書内または無しのいずれかとなる。

引数：

ctlp：PDF ファイルポインタ

uTitle,lppDestName,lpDest,ExpandFlag：アウトライン階層作成関数 2 参照

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.10.6. リモートアウトライン項目作成関数

```
PDFAPI PL_ERROR pl_AddOutline_Remote(hPDF ctlp, const PL_OLTitleTD& uTitle,
        PL_FileActionTypeE fActType, const UCHAR_t* lpFName,const unsigned char* lpDestName,
        PL_DestTD* lpDest,HBOOL ExpandFlag, PL_NewWindowE newWinFlag = PL_NW_NONE);
```

機能：

現在のアウトラインの階層下にもう一つファイル外に宛先をもつアウトライン項目を追加する。

引数：

ctlp：ファイルポインタ

uTitle、fActType、lpFName 、lpDestName、lpDest、ExpandFlag、newWinFlag：

リモートアウトライン階層作成関数参照

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.10.7. アウトライン階層クローズ関数

```
PDFAPI PL_ERROR pl_CloseOutlineLevel(hPDF ctlp)
```

機能：

現在のアウトライン階層を閉じる。

引数：

ctlp：PDF ファイルポインタ

戻り値：

正常終了　　0 を戻す

正常でない終了　　エラーコードを戻す

8.11. 注釈オブジェクト出力関数

注釈（Annotation）はメモ、サウンド、ムービーなどのオブジェクトを PDF 文書のページ上の場所に関連付ける、ページのコンテンツとは独立したオブジェクトである。PDF の注釈にはテキスト注釈、リンク、など多数の注釈が存在するが、PDF Reference Fifth Edtiion では以下のような分類がされている。

1. マークアップ注釈：PDF ドキュメントをマークアップするために使用される注釈で、これは注釈のもつ Contents エントリの使用方法によって、以下の 3 種類に細分化される。
   - 分類 1：Contents エントリが直接、表示用のテキストとなるタイプ

     FreeText
   - 分類 2：通常、その注釈に関連付けされる Popup を持ち、Contents エントリの内容はその Popup 内に表示されるタイプ

     Text、Line、Square、Circle、Polygon(PDF1.5)、PolyLine(PDF1.5)、
     Highlight、Underline、Squiggly(PDF1.4)、StrikeOut、
     Stamp、Ink、FileAttachment、Caret(PDF1.5)
   - 分類 3：Popup を持たないが、Contents エントリにテキストを持つタイプ

     Sound
2. 非マークアップ注釈：上記以外の注釈で、これはさらに、2 種類に細分化されている。
   - 分類 4：Popup 注釈

     Popup 注釈は、通常、上記分類 2 の注釈との組み合わせで使用されるものであり、その場合は組み合わされる注釈の Contents エントリを表示テキストとする。単独で存在した場合は、Popup 注釈自身の Contents エントリの内容を表示する（単独で存在する Popup は Acrobat では作成できない）。
   - 分類 5：Contents エントリを代替テキストとして使用するタイプ

     Link、Movie、Widget、Screen(PDF1.5)、PrinterMark(PDF1.4)、TrapNet、
     WaterMark(PDF1.5)、3D(PDF1.5)

✧　下線の注釈は現在、本ライブラリでは未サポートである。

各注釈についての詳細は PDF 仕様書参照。

すべての注釈において、下記の手順で設定する必要がある。

1. まず pl_Annot_Bgn 関数を呼び出して初期化する。
2. pl_Annot_SetCommData 関数を呼び出して注釈の共通データを設定する。
3. 作成する注釈に固有の関数(たとえば、テキスト注釈であれば、pl_Annot_SetText)を呼び出して、各注釈固有のデータを設定する。
4. 最後に pl_Annot_End()関数を呼び出して注釈処理時に使用されたメモリを解放する。

各マークアップ注釈設定で使用される 2 種類の構造体の定義を以下に示す。

PL_MarkupDataTD

PL_MarkupData 構造体はマークアップ注釈共通の情報を定義する構造体である。

PL_MarkupData 構造体の定義を以下に示す。

```
typedef struct{
    PL_MarkupFlagE        mkFlag;              // フラグ
```

```
        PL_AtPopUpTD        *popUp;         // PopUp 注釈を示す
        UCHAR_t             *T;             // Title 文字列
        float               CA              // 透明度(PDF1.4)
        UCHAR_t*            subj            // 説明(PDF1.5)
    } PL_MarkupDataTD;
```

PL_MarkupFlagE の定義を以下に示す。

```
    typedef enum {
        PL_MF_NONE = 0,
        PL_MF_CA    = 1 << 0        // CA を設定する場合、On とする
    } PL_MarkupFlagE;
```

PL_AtPopUpTD

PL_AtPopUpTD 構造体の定義を以下に示す。

```
    typedef struct{
        PL_RectangleTD      rect;       // Popup の表示位置を指定する
        HBOOL               bOpen;      // Popup をオープン状態とするか否かを指定する
    } PL_AtPopUpTD;
```

bOpen で、注釈が最初開いた状態のまま表示するか否かを指定する。

HTRUE: オープン状態、HFALSE：クローズ状態

rect は Popup の表示位置を指定する。

8.11.1. 注釈初期化関数

```
PDFAPI PL_ERROR pl_Annot_Bgn(hPDF ctlp, PL_AnnotTypeE AType,int pageNo=0
                    #ifdef PLCTRL_TAGGEDPDF
                      PL_MKHandle hMarkedContent=0
                    #endif
)
```

機能：

Atype タイプで指定される注釈タイプの設定の初期化を行う。いずれの注釈もこの関数により初期化を行う必要がある。

引数：

ctlp：PDF ファイルポインタ

AType：設定する Annot のタイプ。詳しくは説明参照。

PageNo：

- pageNo=0 であれば、現在出力中のページに対する注釈設定の開始となる。pl_CreatePage ~ pl_ClosePage 内で使用可能である。
- 0 以外の場合、注釈を設定するページ番号の指定となる(ファイル先頭ページが 1)。

pl_ClosePage()で bAnnotExist=HTRUE としたページだけが指定可能である。

●制限事項：0 以外の pageNo は、既に pl_ClosePage 処理されたページに対して、pl_Annot_SetRemoteLink() により外部 PDF ファイルへのリンクを追加する場合に使用することができる。

hMarkedContent: Marked Content の Handle

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

Atype は注釈のタイプを示す。以下に、定義を示す。

```
typedef enum{
    PL_AT_LINK=0,           // リンク注釈
    PL_ AT_TEXTA,           // テキスト注釈
    PL_ AT_SOUND,           // サウンド注釈
    PL_ AT_MOVIE,           // ムービー注釈
    PL_ AT_RSTAMP,          // ラバースタンプ注釈
    PL_ AT_LINE,            // ライン注釈
    PL_ AT_SQUARE,          // 正方形注釈
    PL_ AT_ACIRCLE,         // 円注釈
    PL_ AT_STRIKEOUT,       // ストライクアウト注釈
    PL_ AT_HIGHLIGHT,       // ハイライト注釈
    PL_ AT_UNDERLINE,       // 下線注釈
    PL_ AT_INK,             // インク注釈
    PL_ AT_FATTACH,         // ファイル添付注釈
    PL_ AT_FTEXT,           // フリーテキスト注釈
    PL_ AT_POPUP,           // ポップアップ注釈
    PL_ AT_SQUIGGLY,        // 波線注釈(PDF1.4)
    PL_ AT_POLYLINE,        // 折線注釈(PDF1.5)
    PL_ AT_POLYGON,         // 多角形注釈(PDF1.5)
    PL_ AT_CARET            // キャレット注釈(PDF1.5)
} PL_AnnotTypeE;
```

8.11.2. 注釈共通データ設定関数

PDFAPI PL_ERROR pl_Annot_SetCommData(hPDF ctlp, PL_CommDataTD* lpAComm)

機能：

注釈共通データを設定する。いかなる注釈にもこの関数で共通データを設定しなくてはならない。

引数：

ctlp： PDF ファイルポインタ

lpAComm： 注釈共通データ構造体へのポインタ。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

注釈共通データ構造体 CommDataTD の定義を以下に示す。

```
typedef struct
{
        PL_CommonFlagE   commFlag;      // 設定フラグ
        PL_RectangleTD   Rect;          // 注釈の位置
        PL_ColorTD       Color;         // 注釈の色
        PL_BorderTD      BS;            // 境界線スタイル
        UCHAR_t          *lpAConents;   // 文字列(主に Popup 用)
        PL_AnnotFlagE    F;             // 注釈のフラグ。デフォルト値は 0。
        PL_HighlightModeE H;            // ハイライトモード
} PL_CommDataTD;
```

commonFlags：設定フラグ。この構造体に設定されているメンバの有無を指定する。

PL_CommonFlagE の定義を以下に示す。

```
  typedef enum {
        PL_CF_NONE     = 0,
        PL_CF_COLOR    = 1 << 0, // カラー指定(Color)有り
        PL_CF_BS       = 1 << 1  // 境界線指定(BS)有り
  } PL_CommonFlagE;
```

Rect：ページ上での注釈の位置を設定する。

Color：注釈の色を設定する。

BS：境界線のスタイルを設定する。BorderTD の定義については共通データ形式を参照。

lpAContents：通常、Popup に表示される文字列であるが、注釈によって異なる。分類別の説明を参照

F： 注釈のフラグであり、注釈の特徴を記述する。PL_AtFlagE の定義を以下に示す。

```
  typedef enum {
        PL_ATF_NONE          = 0,
        PL_ATF_INVISIBLE     = 1 << 0, // 不可視(
        PL_ATF_HIDDEN        = 1 << 1, // 注釈を表示・印刷しない
        PL_ATF_PRINT         = 1 << 2, // 印刷時に注釈を印刷する
        PL_ATF_NOZOOM        = 1 << 3, // ページ倍率にあわせた注釈の拡大縮小を行わない
        PL_ATF_NOROTATE      = 1 << 4, // ページの回転にあわせた回転を行わない
        PL_ATF_NOVIEW        = 1 << 5, // 画面に注釈を表示しない
        PL_ATF_READONLY      = 1 << 6, // マウスのクリックに反応したりしない
        PL_ATF_LOCKED        = 1 << 7, // (PDF 1.4) 削除、位置・サイズの変更禁止
```

// 内容の編集は可

PL_ATF_TOGGLENOVIEW = 1 << 8 //(PDF 1.5) 特定のイベントに対して、

// NoView フラグの解釈を反転

} PL_AnnotFlagE;

H：リンク注釈にのみ有効であり、ハイライトモードを指定する。動作タイプは 4 種であり、注釈領域内でマウスボタンが押されるか、押し続けられるときの視覚効果である。

PL_HighlightModEe の定義を以下に示す。

typedef enum{

PL_HL_INVERT=1,PL_HL_NONE,PL_HL_OUTLINE,PL_HL_PUSH

}PL_HighlightModeE;

PL_HL_INVERT：注釈の境界ボックス内の内容を反転する

PL_HL_NONE：ハイライト無し

PL_HL_OUTLINE：注釈の境界線を反転する。

PL_HL_PUSH：注釈の内容をずらして表示し、押されたように見えるようにする。

PDF/X では、BleedBox 内に注釈の Rect が存在することは許可されない。

注釈のカラーには DeviceRGB を使用しなければならないが、PDF/X-1 は DeviceRGB をサポートしない。このため、PDF/X-1 を設定する場合に、Annotation の設定を禁止する。

8.11.3. 注釈終了関数

PDFAPI void pl_Annot_End(hPDF ctlp)

機能：

Annot を終了する。いかなる注釈でも、終了後必ずこの関数を呼び出すこと。

引数：

ctlp： PDF ファイルポインタ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.11.4. フリーテキスト注釈設定関数

PDFAPI PL_ERROR pl_Annot_SetFText(hPDF ctlp, PL_AtFTextTD* lpSetFText,PL_MarkupDataTD* lpMarkup);

機能：

フリーテキスト注釈を設定する。

引数：

ctlp：PDF ファイルポインタ

lpSetFText：フリーテキスト注釈構造体へのポインタ

lpMarkUp：マークアップ注釈構造体へのポインタ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_AtFTextTD 構造体の定義を以下に示す。

```
typedef struct {
    PL_FontTD           FTextFont;      // フォント情報
    PL_FTAlignmentE     align;          // テキスト配置(PDF1.4)
} PL_AtFTextTD;
```

PL_FontTD 構造体の定義についてはフォント設定関数(pl_SetFont)参照。

PL_FTAlignmentE の定義を以下に示す。

```
typedef enum{
    PL_FAT_LEFT,                // 左揃え
    PL_FAT_CENTER,              // 中央揃え
    PL_FAT_RIGHT                // 右揃え
} PL_FTAlignmentE;
```

8.11.5. テキスト注釈設定関数

PDFAPI PL_ERROR pl_Annot_SetText(hPDF ctlp, PL_AtTextTD* lpAText, PL_MarkupDataTD* lpMarkup)

機能：

テキスト注釈を設定する。

引数：

ctlp：PDF ファイルポインタ

lpAText：テキスト注釈データ構造体へのポインタ

lpMarkUp：マークアップ注釈構造体へのポインタ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_AtTextTD 構造体の定義を以下に示す。

```
typedef struct {
    char * lpIconName;      // テキスト注釈のアイコン名を指定する
} PL_ AtTextTD;
```

PDF にテキスト中借用に事前定義されているアイコン名としては以下が存在する。これを指定する。

Comment, Insert, Note, Paragraph, NewParagrah, Key, Help

8.11.6. ライン注釈設定関数

PDFAPI PL_ERROR pl_Annot_SetLine(hPDF ctlp,PL_AtLineTD* lpSetLine,PL_MarkupDataTD* lpMarkup)

機能：

ライン注釈を設定する。

引数：

ctlp：PDF ファイルポインタ

lpSetLine：ライン注釈構造体へのポインタ

lpMarkUp：マークアップ注釈構造体へのポインタ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_AtLineTD 構造体の定義を以下に示す。

```
typedef struct {
    PL_PointTD     LPoint[2];          // 線の開始点、終点
    PL_ColorTD     IC;                 // 線の端点の内部色,DeviceRGB 固定(PDF1.4)
    PL_LineEndStyleE       LE[2];      // 線の開始点、終点の形状(PDF1.4)
    PL_BorderTD   BS;                  // 破線形状
} PL_AtLineTD;
```

PL_PointTD, PL_ColorTD, PL_BorderTD の定義は共通データ形式を参照。

PL_LineEndStyleE の定義を以下にする。

```
typedef enum{
        PL_LE_None=0,          // 端点形状無
        PL_LE_Square,          // 正方形(内部色有)
        PL_LE_Circle,          // 円(内部色有)
        PL_LE_Diamond,         // ダイアモンド形状(内部色有)
        PL_LE_OpenArrow,       // 開いた矢印(内部色有)
        PL_LE_ClosedArrow,     // 閉じた矢印
        PL_LE_Butt,            // バット(PDF1.5)
        PL_LE_ROpenArrow,      // 逆方向の開いた矢印(PDF1.5)
        PL_LE_RCloseArrow      // 逆方向の閉じた矢印(PDF1.5,内部色有)
} PL_LineEndStyleE;
```

8.11.7. 正方形注釈設定関数

PDFAPI PL_ERROR pl_Annot_SetSquare(hPDF ctlp, PL_AtSquareTD* lpSetSqua,PL_MarkupDataTD* lpMarkup)

機能：

正方形注釈を設定する。

引数：

ctlp：PDF ファイルポインタ

lpSetSqua：正方形注釈構造体へのポインタ

lpMarkUp：マークアップ注釈構造体へのポインタ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_AtSquareTD 構造体の定義を以下に示す。

```
typedef struct {
    PL_ColorTD          IC;     // 内部色、DeviceRGB 固定、指定無は内部を透明(PDF 1.4)
    PL_RectangleTD      RD;     // 内部領域と Rect との差分。
                                // 正の値、かつサイズが Rect を超えないようにする
    (PDF1.5)
    PL_AtBdEffectTD     BE;     // 境界線効果辞書(PDF1.5)
} PL_AtSquareTD;
```

PL_AtBdEffectTD 構造体の定義を以下に示す。

```
typedef struct {
    PL_BENameE S;       // 境界効果を表現する名前
    float      I;       // 0 から 2 の間の効果の強度指定値(S が C の場合のみ有効)
} PL_AtBdEffectTD;
```

PL_BEName の定義を以下に示す。

```
typedef enum{
    PL_BE_S=0,          // 境界線効果なし
    PL_BE_C             // 雲形
} PL_BENameE;
```

8.11.8. 円注釈設定関数

PDFAPI PL_ERROR pl_Annot_SetCirle(hPDF ctlp, PL_AtCircleTD* lpSetCir,PL_MarkupDataTD* lpMarkup)

機能：

円注釈を設定する。

引数：

ctlp：PDF ファイルポインタ

lpSetCir：円注釈構造体へのポインタ

lpMarkUp：マークアップ注釈構造体へのポインタ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_AtCircleTD 構造体の定義を以下に示す。各要素の機能は PL_AtSquareTD に同じ。

```
typedef struct {
```

```
    PL_ColorTD          IC;
    PL_RectangleTD      RD;
    PL_AtBdEffectTD     BE;
} PL_AtCircleTD;
```

8.11.9. 多角形注釈設定関数

PDFAPI PL_ERROR pl_Annot_SetPolygon(hPDF ctlp, PL_AtPolygonTD* lpPolygon,PL_MarkupDataTD* lpMarkup)

機能：

多角形注釈を設定する(PDF1.5)

引数：

ctlp：PDF ファイルポインタ

lpPolygon：多角形注釈構造体へのポインタ

lpMarkUp：マークアップ注釈構造体へのポインタ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_AtPolygonTD 構造体の定義を以下に示す。

```
typedef struct {
    int             vertexNum;      // 多角形の頂点数
    PL_PointTD*     lpVertex;       // 頂点座標列
    PL_BorderTD     BS;             // 境界線種
    PL_ColorTD      IC;     // 内部色。DeviceRGB 固定、指定がなければ内部は透明。
    PL_AtBdEffectTD BE;             // 境界線効果辞書
} PL_AtPolygonTD;
```

PL_AtBdEffectTD は正方形注釈の説明参照

8.11.10. 折線注釈設定関数

PDFAPI PL_ERROR pl_Annot_SetPolyLine(hPDF ctlp, PL_AtPolylineTD* lpPolyLine, PL_MarkupDataTD* lpMarkup)

機能：

折線注釈を設定する(PDF1.5)

引数：

ctlp：PDF ファイルポインタ

lpPolyline：折線注釈構造体へのポインタ

lpMarkUp：マークアップ注釈構造体へのポインタ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_AtPolyLineTD 構造体の定義を以下に示す。

```
typedef struct {
    int             vertexNum;      // 折線の頂点数
    PL_PointTD*     lpVertex;       // 頂点座標列
    PL_LineEndStyle LE[2]           // 開始点、終点の形状
    PL_BorderTD     BS;             // 境界線種
    PL_ColorTD      IC;             // 内部色。DeviceRGB 固定、指定がなければ内部は透明。
} PL_AtPolylineTD;
```

PL_AtBdEffectTD は正方形注釈の説明参照

8.11.11. ハイライト注釈設定関数

PDFAPI PL_ERROR pl_Annot_SetHighlight(hPDF ctlp, PL_AtHighLTD* lpHlighLight, PL_MarkupDataTD* lpMarkup)

機能：

ハイライト注釈を設定する。

引数：

ctlp：PDF ファイルポインタ

lpHigh：ハイライト注釈構造体へのポインタ

lpMarkUp：マークアップ注釈構造体へのポインタ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_AtHighLTD 構造体の定義を以下に示す。

```
typedef struct{
    int                 quadPointNum;   // quadpoint の四辺形の数
    PL_QuadPointTD*     lpQuadPoints;   // quadpoint の座標値
} PL_AtHighLTD;
```

PL_QuadPointTD 構造体の定義を以下に示す。

```
typedef struct{
    PL_PointTD;   //bgnBottom,endBottom,endTop,beginTop; //quadpoints の 4 点
} PL_QuadPointTD;
```

lpQuadPoints でハイライトを配置する四辺形の座標 x1 y1 x2 y2 x3 y3 x4 y4 を指定する。頂点の指定順序は反時計回りであり、(x1,y1) ⇒ (x2,y2)を結ぶ辺がテキスト方向となるように指定する。

8.11.12. 下線注釈設定関数

PDFAPI PL_ERROR pl_Annot_SetUnderline(hPDF ctlp, PL_AtUnderLTD* lpUnderLine, PL_MarkupDataTD* lpMarkup)

機能：

下線注釈を設定する。

引数：

ctlp：PDF ファイルポインタ

lpUnderl：下線注釈構造体ポインタ

lpMarkUp：マークアップ注釈構造体へのポインタ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_AtUnderLTD 構造体の定義を以下に示す。

```
typedef struct{
    int                 quadPointNum;    // quadpoint の四辺形の数
    PL_QuadPointTD*     lpQuadPoints;    // quadpoint の座標値
} PL_AtUnderLTD;
```

lpQuadPoints についてはハイライト注釈設定関数参照。下線注釈では begBottom から endBottom に下線が表示される。

8.11.13. 波線注釈設定関数

PDFAPI PL_ERROR pl_Annot_SetSquiggly(hPDF ctlp, PL_AtSquigglyTD* lpSquiggly, PL_MarkupDataTD* lpMarkup)

機能：

下線注釈を設定する。

引数：

ctlp：PDF ファイルポインタ

lpSquiggly：波線注釈構造体ポインタ

lpMarkUp：マークアップ注釈構造体へのポインタ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_AtUnderLTD 構造体の定義を以下に示す。

```
typedef struct{
    int                 quadPointNum;    // quadpoint の四辺形の数
    PL_QuadPointTD*     lpQuadPoints;    // quadpoint の座標値
} PL_AtSquigglyLTD;
```

lpQuadPoints についてはハイライト注釈設定関数参照。波線注釈では begBottom から endBottom に波線が表示される。

8.11.14. ストライクアウト注釈設定関数

PDFAPI PL_ERROR pl_Annot_SetStrike(hPDF ctlp, PL_AtStrikeTD* lpStrikeOut,PL_MarkupDataTD* lpMarkup)

機能：

ストライクアウト注釈を設定する。

引数：

ctlp：PDF ファイルポインタ

lpStrike：ストライクアウト注釈構造体ポインタ

lpMarkUp：マークアップ注釈構造体へのポインタ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_AtStrikeTD 構造体の定義を以下に示す。

```
typedef struct{
    int                 quadPointNum;    // quadpoint の四辺形の数
    PL_QuadPointTD*     lpQuadPoints;    // quadpoint の座標値
} PL_AtStrikeTD;
```

lpQuadPoints についてはハイライト注釈設定関数参照。

ストライクアウト注釈では begBottom と begTop の中央部から endBottom と endTop 中央部に取り消し線が表示される。

8.11.15. ラバースタンプ注釈設定関数

PDFAPI PL_ERROR pl_Annot_SetStamp(hPDF ctlp,PL_AtRStampTD* lpSetStamp, PL_MarkupDataTD* lpMarkup)

機能：

ラバースタンプ注釈を設定する。

引数：

ctlp：PDF ファイルポインタ

lpSetStamp：ラバースタンプ注釈構造体へのポインタ

lpMarkUp：マークアップ注釈構造体へのポインタ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_AtRStampTD 構造体の定義を以下に示す。

typedef struct {

char *lpIconName; // スタンプのアイコン名

} PL_AtRStampTD;

lpIconName にスタンプ形状のアイコン名を指定する。PDF では事前定義のアイコン名として以下が定義されている。これを指定する。

Approved(承認済)、AsIs(未変更)、Confidential(親展)、Departmental(内部用)、Draft(草稿)、Experimental(試用)、Expired(失効)、Final(最終)、ForComment(推敲待)、ForPublicRelease(公開用)、NotApproved(却下)、NotForPublicRelease(非公開)、Sold(売却済)、TopSecret(極秘)

8.11.16. キャレット注釈設定関数

PDFAPI PL_ERROR pl_Annot_SetCaret(hPDF ctlp, PL_AtCaretTD* lpCaret,PL_MarkupDataTD* lpMarkup)

機能：

キャレット注釈を設定する。

引数：

ctlp：PDF ファイルポインタ。

lpCaret：キャレット注釈データ構造体へのポインタ

lpMarkUp：マークアップ注釈構造体へのポインタ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

キャレット注釈データ構造体の定義を以下に示す。

typedef struct {

PL_RectangleTD RD; // キャレットを包含する矩形と Rect との差分を
// 記述する 4 数値。left,top.right,bottom の差(PDF1.5)

PL_SymbolNameE Sy; // キャレットとして使用されるシンボル名
// デフォルト値：None

} PL_AtCaretTD;

PL_SymbolNameE の定義を以下に示す。

typedef enum{

PL_SY_None=0, // キャレットでシンボルを使用しない

PL_SY_P // 新段落シンボル(“¶”)

} PL_SymbolNameE;

8.11.17. インク注釈設定関数

PDFAPI PL_ERROR pl_Annot_SetInk(hPDF ctlp,PL_AtInkTD* lpSetInk,PL_MarkupDataTD*

lpMarkup)

機能：

インク注釈を設定する。

引数：

ctlp：PDF ファイルポインタ

lpSetInk：インク注釈構造体へのポインタ

lpMarkUp：マークアップ注釈構造体へのポインタ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_AtInkTD 構造体の定義を以下に示す。

```
typedef struct {
    int             lNum;       // 線の数
    PL_OneLineTD*   lpLine;     // 線へのポインタ列
} PL_AtInkTD;
```

PL_OneLineTD 構造体の定義を以下に示す。

```
typedef struct{
    int             PointNum;   // ポイントの数
    PL_PointTD*     lpPoint;    //  1 本の線
} PL_OneLineTD;
```

PL_OneLineTD が 1 本のパスに対応し、これを複数指定することでフリーハンドの線を出力する。

8.11.18. ファイル添付注釈設定関数

PDFAPI PL_ERROR pl_Annot_SetAttFile(hPDF ctlp, PL_AtFAttcTD* lpSetFileAt,PL_MarkupDataTD* lpMarkup)

機能：

ファイル添付注釈を設定する。

引数：

ctlp：PDF ファイルポインタ。

lpSetFileAt：ファイル添付注釈データ構造体へのポインタ

lpMarkUp：マークアップ注釈構造体へのポインタ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

ファイル添付注釈データ構造体の定義を以下に示す。

```
typedef struct {
```

```
    char *lpFileName;           // 添付ファイル名(ASCII)
    UCHAR_t* lpUFileName;       // 添付ファイル名(Unicode)
    char *lpIconName;           // アイコン種別
    PL_DateTD createDate;       // 作成日
    PL_DateTD modDate;          // 更新日
} PL_AtFAttcTD;
```

PDF では添付ファイル注釈用に事前定義されたアイコン名として以下が存在する。lpIconName にはこれを指定する。

PushPin(添付)、Graph(グラフ)、PaperClip(クリップ)、Tag(タグ)

8.11.19. サウンド注釈設定関数

PDFAPI PL_ERROR pl_Annot_SetSound(hPDF ctlp, PL_AtSoundTD* lpSetSound, PL_MarkupDataTD* lpMarkup)

機能：

サウンド注釈を設定する。

引数：

ctlp：PDF ファイルポインタ。

lpSetSound：サウンド注釈構造体へのポインタ

lpMarkUp：マークアップ注釈構造体へのポインタ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_AtSoundTD 構造体の定義を以下に示す。

```
typedef struct {
    PL_SoundTypeE       soundType;      // 内部ストリーム、外部ファイルの種別
    PL_SoundFileTD      soundFile;      // 外部ファイル用サウンド定義
    PL_SoundStreamTD    soundStream;    // 内部ストリーム用サウンド定義
    char                *lpIconName;    // アイコン名称
} PL_AtSoundTD;

typedef enum{
    PL_ST_INNERSTREAM,      // サウンドデータをファイル内ストリームとする
    PL_ST_EXTERANLFILE      // サウンドデータをファイル外に置く
} PL_SoundTypeE;
typedef struct {
    char *lpESFName;            // 外部サウンドファイル名(ASCII)
    UCHAR_t* lpUESFName;        // 外部サウンドファイル名(Unicode)
} PL_SoundFileTD;
```

```
typedef struct {
    PL_FilterE      filterType;     // サウンドファイルフィルタ
    float           R;              // サンプリングレート
    int             C;              // サウンドチャネル数(Default value： 1)
    int             B;              // チャネルあたりのサンプル値のビット数
                                    // Default value： 8
    PL_Sound_EE E;                  // サンプルデータのエンコーディング形式
                                    // Default value： PL_SOUND_E_RAW.
    PL_FilterE      CO;             // サウンドのサンプルデータの圧縮形式
    char*           SStream;        // サウンドストリーム
    int             Length;         // ストリーム長
} PL_SoundStreamTD;

typedef enum{
    PL_FILTER_NONE,                 // フィルタ無し
    PL_FILTER_FLATEDECO,            // Flate フィルタ
    PL_FILTER_ASCIIHEXDECO,         // ASCII Hex フィルタ
    PL_FILTER_ASCII85DECO,          // ASCII 85 フィルタ
    PL_FILTER_LZWDECOD,             // LZW フィルタ
    PL_FILTER_RUNLENGTHDECO,        // RunLength フィルタ
    PL_FILTER_DCTDECO,              // DCT フィルタ
    PL_FILTER_CCITTFAXDECO          // CCITTFax フィルタ
} PL_FilterE;

typedef enum{
    PL_SOUND_E_RAW=0,  // 0 から 2^B-1 までの範囲の指定無しまたは符号無しの値
    PL_SOUND_E_SIGNED,      // 2 の補数の値
    PL_SOUND_E_MULAW,       // μ-law 形式でエンコードされたサンプル
    PL_SOUND_E_ALAW         // A-law 形式でエンコードされたサンプル
} PL_Sound_EE;
```

PDF ではサウンド注釈用に事前定義されたアイコン名として以下が存在する。lpIconName にはこれを指定する。

Speaker(スピーカ)、Mic(マイクロフォン)

8.11.20. ポップアップ注釈設定関数

PDFAPI PL_ERROR pl_Annot_SetPopUp(hPDF ctlp,PL_AtPopUpTD* lpSetPopUp,

PL_PopUpContentTD* lpPopUpContents)

機能：

ポップアップ注釈を設定する。

引数：

ctlp：PDF ファイルポインタ

lpSetPopUp：ポップアップ注釈構造体へのポインタ

lpPopUpContents：ポップアップのタイトル

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_AtPopUpTD については、本項(注釈オブジェクト出力)先頭を参照

PL_PopUpContent の構造体の定義を以下に示す。

```
typedef struct{
    UCHAR_t*    T;                  // ポップアップのタイトル
} PL_PopUpContentTD;
```

通常のポップアップはテキスト注釈、ライン注釈など、他の注釈と組み合わせて使用されるため、単独でこの関数を使用することはない。この関数の使用は推奨されない。

8.11.21. ムービー注釈設定関数

PDFAPI PL_ERROR pl_Annot_SetMovie(hPDF ctlp, PL_AtMovieTD* lpSetMovie)

機能：

ムービー注釈を設定する

引数：

ctlp：PDF ファイルポインタ。

lpSetMovie：ムービー注釈構造体へのポインタ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_AtMovieTD 構造体の定義を以下に示す。

```
typedef struct{
    PL_SCharasTD        SCharas;        // 静的特性
    PL_ACharasTD        ACharas;        // 動的特性
    HBOOL               bAflag;         // 注釈アクティブ時ムービー起動指定
                                        // HTRUE：デフォルトパラメータで再生
                                        // HFALSE：再生しない
    HBOOL               bControl        // HFALSE：Aflag 有効
```

```
                                        // HTRUE：Aflag 無効
} PL_AtMovieTD;
```

SCharas はムービーの静的特性を記述するためのもので、必ず指定する必要がある ACharas はムービー再生中の動的特性を記述するためのもので、必要がある場合にのみ指定する。ACharas を使用する場合、Control を true にする。

PL_SCharasTD 構造体の定義を以下に示す。

```
typedef struct{
    char        *lpMovFName;        // 指定するファイル(ASCII)
    UCHAR_t     *lpUMovFName;       //指定するファイル(Unicode)
    PL_AspectTD Aspect;             // 境界枠の幅と高さ
    int         Rotate;             // 回転角度(90 の倍数、時計回り)
    HBOOL       bPoster;            // Poster イメージを表示できるか否か
                                    //  （イメージストリームは未サポート）
} PL_SCharasTD;
```

PL_AspectTD 構造体の定義を以下に示す。

```
typedef struct{
    float       horiz;              // ムービーの境界枠の幅
    float       vert;               // ムービーの境界枠の高さ
} PL_AspectTD;
```

bControl に HTRUE を設定すると Aflag 値が有効になる。この場合構造体 ACharas を設定する必要はない。bAflag 値はマウスクリック時に、ムービーファイルを再生するか否かを指定する。デフォルトは HTRUE である。

bControl に HFALSE を設定すると、Aflag 値が無効になり、代わりに ACharas 構造体が参照される。

PL_ACharasTD 構造体の定義を以下に示す。

```
typedef struct{
    HBOOL           bSCtrl;         // コントロールバーを表示するか否か
    PL_MovieModeE   Mode;           // 再生 mode(“Once”、”Open”、”Repeat”)
    HBOOL           bSync;          // 同期か否か
    float           Start;          // ムービーの開始時間
    float           Duration;       // ムービーの持続期間
    float           Rate;           // ムービー再生の初期スピード
    float           Volume;         // 初期音量、 –1 から 1 まで
    PL_FWScaleTD    FWScale;        // 表示の倍率
    PL_FWPosTD      FWPos;          // ウィンドウの相対位置
} PL_ACharasTD;
```

PL_FWScaleTD と PL_FWPosTD の定義を以下に示す。

```
typedef struct{
```

```
    float           numerator;      // 表示の倍率の numerator
    float           denominator;    // 表示の倍率の dominator
} PL_FWScaleTD;

typedef struct{
    float           horiz;          // ウィンドウ相対位置の水平座標
    float           vert;           // ウィンドウ相対位置の垂直座標
} PL_FWPosTD;

typedef enum{
    PL_MM_ONCE,             // 1 回再生して停止
    PL_MM_OPEN,             // 再生後、コントロールバーを開いたまま表示
    PL_MM_REPEAT,           // 停止されるまで繰り返し再生
    PL_MM_PALINDROME        // 停止まで順方向・逆方向連続再生
} PL_MovieModeE;
```

8.11.22. リンク注釈(GoTo)設定関数

PDF ファイル内部へのリンクを行う GoTo アクションを使用したリンク注釈の設定を行う。この注釈を設定する場合、リンク元の設定関数とリンク先の設定関数を呼び出す必要がある。（ファイル内部リンクは、リンク元とリンク先が関係し、両者を同時には出力できない場合があるため、関数を分離してある）

リンク元設定関数

PDFAPI PL_ERROR pl_Annot_SetLinkSrc(hPDF ctlp, const UCHAR_t* lpALinkSrc)

機能：

GoTo アクションを使用したリンク注釈のリンク元を設定する。

引数：

ctlp：PDF ファイルポインタ

lpALinkSrc：リンク宛先名（リンク先設定関数で使用する名称を指定する）

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

リンク先設定関数

PDFAPI PL_ERROR pl_Annot_SetLinkDest(hPDF ctlp, PL_AtLinkDestTD* lpALinkDest)

機能：

GoTo アクションを使用したリンク注釈のリンク先を設定する。

引数：

ctlp：PDF ファイルポインタ

lpALinkDest：リンク先データ構造体へのポインタ。リンク先の情報を格納する。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_ AtLinkDestTD 構造体の定義を以下に示す。

```
typedef struct
{
    UCHAR_t     *lpLinkStr      // リンク宛先名(リンク元設定関数で使用する名称を
    指定する)
    PL_ DestTD  Dest;           // 宛先
} PL_ AtLinkDestTD;
```

PL_ DestTD については、共通データ形式の記載参照。PL_DestTD 内にはページ番号を指定する PgNo が存在するが、本関数ではこのメンバを参照しない。本関数呼び出し時に処理中のページに対する指定とみなすことに注意。

pl_Annot_SetLinkDest 使用時、同じリンク宛先名を指定された未処理の（リンク先を割り当てていない）pl_Annot_SetLinkSrc が存在する場合、そのリンク元をこのリンク先に結合する。なければ、宛先をリンク元の指定に備えて、内部で記録する。

pl_Annot_SetLinkSrc 使用時、同じリンク宛先名を指定された pl_Annot_SetLinkDest があれば、その宛先にリンクする。存在しない場合は、このリンク元は保留される。最後まで宛先が指定されない場合、[/XYZ null null null]（何もしないリンク）が設定される。リンク元が指定されないリンク先が残った場合、そのリンク先は文書には出力されず、無視される。

8.11.23. リンク注釈(GoTo)設定関数 2

```
PDFAPI PL_ERROR pl_Annot_SetLocalLink(hPDF ctlp, const unsigned char* lpDestName,
                                      PL_DestTD* lpDest);
```

機能：

GoTo アクションを使用したリンク注釈を設定する。前の 2 つの関数でファイル内部リンクを設定する方法と以下の点が異なる。

- １つだけの関数でファイル内部のリンクを設定できる
- 名前付き宛先により宛先を指定することができる

引数：

ctlp：PDF ファイルポインタ

lpDestName,lpDest：名前または宛先

- null ではないものが有効である
- 二つともに null ではないなら、lpDestName を使用する
- 二つともに null であるなら、宛先指定の無い Link を作成する

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

名前付き宛先は pl_SetNames で定義する

8.11.24. リンク注釈(Launch/URI)設定関数

PDFAPI PL_ERROR pl_Annot_SetLinkOther(hPDF ctlp, const UCHAR_t* LinkFName,
PL_LinkTypeE LType,PL_NewWindowE newWinFlag = PL_NW_NONE)

機能：

外部ファイルの起動を行う Launch アクションを使用したリンク注釈、または Web 上のファイルにリンクする URI アクションを使用したリンク注釈の設定を行う。

LType によっていずれのリンクとするかを指定する。

引数：

ctlp：PDF ファイルポインタ

LinkFName：リンク宛先のファイル名

LType：WEB 上のファイル、またはその他のファイル

NewWinFlag(optional)： 新しいウィンドウで目的ファイルを開くか否かを指定する。

Launch アクションの場合のみ有効（URI アクションには NewWindow は無い）

リモートアウトライン階層作成関数参照

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

LTypee：リンクのアクションのタイプを指定する。

```
typedef enum{
    PL_LT_INFILE=1,          // (使用禁止)
    PL_ LT_OTHERFILE,        // Launch アクション
    PL_ LT_WEBFILE           // URI アクション
} PL_LinkTypeE;
```

PL_LineTypeE の定義は前述の説明を参照のこと。

8.11.25. リンク注釈(GoToR)設定関数

PDFAPI PL_ERROR pl_Annot_SetRemoteLink(hPDF ctlp, const UCHAR_t* lpFName,
const unsigned char* lpDestName,
PL_DestTD* lpDest, PL_NewWindowE newWinFlag = PL_NW_NONE)

機能：

外部 PDF ファイル内の宛先へのリンク注釈の設定を行う。

引数：

ctlp：PDF ファイルポインタ

lpFName： Link 先の PDF ファイル名（null なら、宛先指定の無い Link を作成する）

lpDestName,lpDest：名前または宛先

- null ではないものが有効である;
- 二つともに null ではないなら、lpDestName を使用する
- 二つともに null であるなら、宛先指定の無い Link を作成する

NewWinFlag(optional)：新しいウィンドウで目的ファイルを開くか否かを指定する。リモートアウトライン階層作成関数参照

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

名前付き宛先はリンク先の PDF ファイルに設定されているものを指定する

既に pl_ClosePage されたページに対して、pl_AnnotBgn で pageNo を指定して、注釈を付加することができる。この場合、pl_ClosePage 時に bAnnotExist= HTRUE としておく必要がある。

8.11.26. アクション指定のリンク注釈

```
PDFAPI PL_ERROR pl_Annot_SetLinkByActHandle (hPDF ctlp,
                                             const PL_ActionHandle &hAction)
```

機能：

アクションが指定されたリンク注釈の設定を行う。

引数：

ctlp：PDF ファイルポインタ

hAction：アクションのハンドル

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

hAction は後述のアクション生成関数により作成されるアクションのハンドルである。

8.11.27. アクション指定のリンク注釈 2

```
PDFAPI PL_ERROR pl_Annot_SetLinkDestByActHandle (hPDF ctlp,
                                                 const PL_AtLinkDestActionTD  atLinkDestAction)
```

機能：

アクションが指定されたリンク注釈の設定を行う。

引数：

ctlp：PDF ファイルポインタ

atLinkDestAction： リンク時のアクションを指定する。

定義を以下に示す。

```
typedef struct {
    UCHAR_t*            lpLinkStr;      // リンク元と宛先を対応付ける文字列
    PL_ActionHandle     hAction;        // アクションのハンドル
} PL_AtLinkDestActionTD;
```

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

これは、前述の pl_Annot_SetLinkData 関数のアクション指定版である。pl_Annot_SetLinkSrc で指定されたリンク元に対応するアクションを、pl_Annot_SetLinkDest 同様に、別のタイミングで設定する必要がある場合に使用する。

8.12. PDF インポート

他の PDF ファイルのページコンテンツをそのまま、現在作成中のページの矩形領域に埋め込むことを本ライブラリではインポートと呼ぶ。この時に呼出す関数のシーケンスを以下に示す。

pl_ImpPdf_Initial( ) ⇒pl_ImpPdf_Check( )

⇒ pl_ImpPdf_GetPgCount( )、pl_ImpPdf_GetVer( )、pl_ImpPdf_GetSecurity ( )

⇒ pl_ImpPdf_SetPgNo( )

⇒ pl_ImpPdf_GetPgSize ( )、pl_ImpPdf_GetPgRotate ()

⇒ pl_ImpPdf_Do( )

⇒ pl_ImpPdf_Finish ( )

以下の呼び出しは不要であれば省略可能である。

pl_ImpPdf_GetPgCount (),pl_ImpPdf_GetVer (),pl_ImpPdf_GetSecurity (),

pl_ImpPdf_GetPgSize (),pl_ImpPdf_GetPgRotate ()

pl_ImpPdf_SetBoxMode( )、pl_ImpPdfGetPgSizeEx( )、pl_GetPgBox

以下は省略不可である。

pl_ImpPdf_Initial()、pl_ImpPdf_SetPgNo ()、pl_ImpPdf_Do () または pl_ImpPdf_DoEx ()

pl_ImpPdf_Finish ()

8.12.1. PDF インポート初期化関数

PDFAPI PL_ERROR pl_ImpPdf_Initial(hPDF ctlp,const char* pfName, hIMPPDF* plpImpPdf)

機能：

他の PDF ファイルのインポート処理用の初期化作業-メモリ要求を行う。

引数：

ctlp ： PDF ファイルのポインタ

pfName：インポートされる PDF ファイルの名称

plpImpPdf： インポートされる PDF ファイルデータを格納するポインタアドレス

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.12.2. PDF インポート初期化関数(Stream 版)

PDFAPI PL_ERROR pl_ImpPdf_Initial_Stream(hPDF ctlp,
std::istream& ismImpPdf,hIMPPDF* plpImpPdf, const char* pfName = NULL)

機能：

他の PDF ストリームのインポート処理用の初期化作業-メモリ要求を行う。

引数：

ctlp ： PDF ファイルのポインタ

ismImpPdf : インポートされる PDF ストリーム

plpImpPdf： インポートされる PDF ストリームデータを格納するポインタアドレス

pfName： 入力ストリームのファイル名（エラー表示用）

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.12.3. インポート PDF ファイルテスト関数

PDFAPI PL_ERROR pl_ImpPdf_Check(hPDF ctlp, hIMPPDF lpImpPdf)

機能：

インポート可能な PDF ファイルか否かを判定する

引数：

ctlp ： PDF ファイルポインタ

lpImpPdf : インポートされる PDF ファイルのデータポインタ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PDF に何らかのセキュリティ設定が施されている場合、および、現在出力中の PDF ファイルよりバージョンが高い場合はインポートできない。

8.12.4. インポート PDF ファイルの頁数取得関数

PDFAPI PL_ERROR pl_ImpPdf_GetPgCount(hPDF ctlp, hIMPPDF lpImpPdf, int& pgCount)

機能：

インポートされる PDF ファイルの総頁数を取得する。

引数：

ctlp ： PDF ファイルポインタ

lpImpPdf ： インポートされる PDF ファイルのデータポインタ

pgCount ： インポートされる PDF ファイルの総頁数

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.12.5. インポート PDF ファイルのバージョン取得関数

PDFAPI PL_ERROR pl_ImpPdf_GetVer(hPDF ctlp, hIMPPDF lpImpPdf,PL_Version& PDFVer)

機能：

インポートされる PDF ファイルのバージョン情報を取得する

引数：

ctlp ： PDF ファイルポインタ

lpImpPdf ： インポートされる PDF ファイルのデータポインタ

PDFVer ： インポートされる PDF ファイルのバージョン情報

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.12.6. インポート PDF ファイルのセキュリティ情報取得関数

PDFAPI PL_ERROR pl_ImpPdf_GetSecurity(hPDF ctlp, hIMPPDF lpImpPdf,
PL_ImpPdf_EncryptInfoTD & encryptInfo)

機能：

インポートされる PDF ファイルのセキュリティ情報を取得する。

引数：

ctlp ： PDF ファイルポインタ

lpImpPdf ： インポートされる PDF ファイルのデータポインタ

encryptInfo： インポートされる PDF ファイルのセキュリティ情報

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_ImpPdf_EncryptInfoTD の定義を以下に示す。

```
typedef struct
{
    HBOOL hasUPassWord;          // ユーザパスワードが設定されている
    HBOOL hasOPassWord;          // オーナパスワードが設定されている
    PL_PermissionE Permission;   // 権限設定フラグ
    PL_RevisionE Revision;       // 標準セキュリティハンドラのレビジョン
} PL_ImpPdf_EncryptInfoTD;
```

8.12.7. インポート PDF のページ設定関数

PDFAPI PL_ERROR pl_ImpPdf_SetPgNo(hPDF ctlp, hIMPPDF lpImpPdf, int pgNo)

機能：

インポートされる PDF ファイルのページを指定する

引数：

ctlp ： PDF ファイルポインタ

lpImpPdf : インポートされる PDF ファイルのデータポインタ

pgNo : インポートされる PDF ファイルの頁番号（先頭ページを 1 とする）

戻り値 :

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.12.8. インポート PDF の用紙サイズ取得関数

PDFAPI PL_ERROR pl_ImpPdf_GetPgSize(hPDF ctlp, hIMPPDF lpImpPdf, float& pgW,float& pgH)

機能 :

インポートされる PDF ファイルの指定頁の用紙サイズを取得する。

引数 :

ctlp ： PDF ファイルポインタ

lpImpPdf : インポートされる PDF ファイルのデータポインタ

pgW,pgH : インポートされる PDF ファイルの指定頁の幅と高さを戻す

戻り値 :

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明 :

ここで戻す用紙サイズは pl_ImpPdf_SetBoxMode で指定する。指定されていない場合、PDF の CropBox と MediaBox の共通領域領域である。

8.12.9. インポート PDF の回転角度取得関数

PDFAPI PL_ERROR pl_ImpPdf_GetPgRotate(hPDF ctlp, hIMPPDF lpImpPdf,int& rAngle)

機能 :

インポートされる PDF ファイルの指定頁の回転角度を取得する。

引数 :

ctlp ： PDF ファイルポインタ

lpImpPdf : インポートされる PDF ファイルのデータポインタ

rAngle : インポートする PDF ファイルの指定頁の回転角度を取得する

戻り値 :

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.12.10. PDF インポート実行関数

PDFAPI PL_ERROR pl_ImpPdf_Do(hPDF ctlp, hIMPPDF lpImpPdf,float x,float y, float w,float h)

機能 :

現在出力中のページの指定位置点に他の PDF ファイルの 1 頁をインポートする

引数 :

ctlp ： PDF ファイルポインタ

lpImpPdf： インポートさる PDF ファイルのデータポインタ

x,y： インポートする PDF の左下端のカレントページ上での表示位置の座標

w,h： インポートする PDF を表示する領域のカレントページ上での幅と高さ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.12.11. PDF インポート終了関数

PDFAPI void pl_ImpPdf_Finish(hPDF ctlp, hIMPPDF* plpImpPdf)

機能：

PDF ファイルのインポートで使用したメモリを解放など

引数：

ctlp ： PDF ファイルポインタ

plpImpPdf： インポートされる PDF ファイルデータを格納するポインタアドレス

戻り値：

無し

8.12.12. インポート PDF のページ境界取得関数

PDFAPI PL_ERROR pl_ImpPdf_GetPgBox(hPDF ctlp, hIMPPDF lpImpPdf,
PL_ImpPdf_BoxModeE boxMode,PL_RectangleTD& pgBox)

機能：

インポートする PDF ファイルの各種ページ境界を取得する

引数：

ctlp ： PDF ファイルポインタ

lpImpPdf：インポートする PDF ファイルデータを格納するポインタアドレス

boxMode：取得する境界ボックスを指定する

pgBox：境界ボックスの値が戻る

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

境界ボックスを以下から指定する。

```
typedef enum{
    PL_ IMPBM _SUBMC=0,      // クロップボックスとメディアボックスの共通領域
    PL_ IMPBM _MEDIA,        // メディアボックス
    PL_ IMPBM _CROP,         // クロップボックス
    PL_ IMPBM _BLEED,        // ブリードボックス
    PL_ IMPBM _TRIM,         // トリムボックス
    PL_ IMPBM _ART,          // アートボックス
```

} PL_ImpPdf_BoxModeE;

8.12.13. インポート PDF のページ境界指定関数

PDFAPI PL_ERROR pl_ImpPdf_SetBoxMode(hPDF ctlp, hIMPPDF lpImpPdf,PL_ImpPdf_BoxModeE boxMode)

機能：

インポートする PDF ファイルの各種ページ境界からインポート対象を指定する

引数：

ctlp ： PDF ファイルポインタ

lpImpPdf： インポートする PDF ファイルデータを格納するポインタアドレス

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

pl_ImpPdf_GetPgSize が戻す（インポート対象とする）境界ボックスを指定する。この関数が使用されない場合、CropBox と MediaBox の共通領域がインポート対象となる。

8.12.14. 領域指定付き PDF インポート実行関数

PDFAPI PL_ERROR pl_ImpPdf_DoEx(hPDF ctlp, hIMPPDF lpImpPdf,float x,float y, float w,float h, float x2, float y2, float w2, float h2)

機能：

現在出力中のページの x、y、w、h で指定される領域に、インポートする PDF ファイルの指定されたページ境界の領域の x2,y2,w2,h2 で指定される領域をインポートする。

PL_ImpPdf_DoEx の領域指定機能の追加バージョンである。

引数：

ctlp ： PDF ファイルポインタ

lpImpPdf： インポートする PDF ファイルデータを格納するポインタアドレス

x,y： インポートする PDF の左下端のカレントページ上での表示位置の座標

w,h： インポートする PDF を表示する領域のカレントページ上での幅と高さ

x2,y2： インポートするページの表示対象とする領域の左下端の座標
インポートする PDF の pl_ImpPdf_SetBoxMode で指定した領域の左下端を原点とする座標系で指定する

w2,h2： インポートするページの表示対象とする領域の幅と高さ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.12.15. インポート PDF のタグ情報検査関数

PDFAPI PL_ERROR pl_ImpPdf_IsTaggedPDF(hPDF ctlp, hIMPPDF lpImpPdf, HBOOL& bTaggedPdf)

機能：

インポートする PDF がタグ付 PDF であるか否かを検査する。

引数：

ctlp ： PDF ファイルポインタ

lpImpPdf： インポートされる PDF ファイルデータを格納するポインタアドレス

bTaggedPdf : インポートされる PDF ファイルがタグ付 PDF か否かを示す

HTRUE	タグ付 PDF
HFALSE	タグ付 PDF ではない

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.12.16. インポート PDF の OutputIntent の数の取得関数

PDFAPI void pl_ImpPdf_GetOutputIntentCount(hPDF ctlp, hIMPPDF lpImpPdf, int& opiCount)

機能：

インポートされる PDF ファイルから OutputIntent 的数量を取得する。

引数：

ctlp ： PDF ファイルポインタ

lpImpPdf： インポートされる PDF ファイルデータを格納するポインタアドレス

opiCount : インポートされる PDF ファイルが使用の OutputIntent 的数量.

戻り値：

無し

8.12.17. インポート PDF の OutputIntent 取得関数

PDFAPI PL_ERROR pl_ImpPdf_GetOutputIntent(hPDF ctlp, hIMPPDF lpImpPdf,
PL_OPHandle& hOutputIntent, int opiNo=1)

機能：

インポートされる PDF ファイルから OutputIntent を取得する。

引数：

ctlp ： PDF ファイルポインタ

lpImpPdf : インポートされる PDF ファイル構造体ポインタ。

hOutputIntent : インポートされる PDF ファイルが使用する OutputIntent

OpiNo : 取得する OutputIntent の番号を指定する(1Origin で、pl_ImpPdf_GetOutputIntentCount で習得した数の範囲で指定のこと

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.12.18. インポート PDF のファイル ID 取得関数

PDFAPI PL_ERROR pl_ImpPdf_GetFileID(hPDF ctlp,hIMPPDF lpImpPdf, std::string& fileID)

機能：

インポートされる PDF ファイルからファイル ID を取得する。

引数：

ctlp ： PDF ファイルポインタ

lpImpPdf：インポートされる PDF ファイル構造体ポインタ。

fileID：インポートされる PDF のファイル ID

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.12.19. インポート PDF の有効注釈設定関数

PDFAPI void pl_SetAnnotsEnableOpt(hPDF ctlp,PL_AnnotsEnableE enableAnnots)

機能：

インポートされる PDF の有効にする注釈を設定する。

引数：

ctlp ： PDF ファイルポインタ

enableAnnots：有効にする注釈.。複数指定は論理輪で行う。

```
typedef enum{
    PL_ANNOTSENABLE_NONE          =   0,
    PL_ANNOTSENABLE_LINK          =   1 << 0,
    PL_ANNOTSENABLE_TEXT          =   1 << 1,
    PL_ANNOTSENABLE_SOUND         =   1 << 2,
    PL_ANNOTSENABLE_MOVIE         =   1 << 3,
    PL_ANNOTSENABLE_STAMP         =   1 << 4,
    PL_ANNOTSENABLE_LINE          =   1 << 5,
    PL_ANNOTSENABLE_SQUARE        =   1 << 6,
    PL_ANNOTSENABLE_CIRCLE        =   1 << 7,
    PL_ANNOTSENABLE_STRIKEOUT     =   1 << 8,
    PL_ANNOTSENABLE_HIGHLIGHT     =   1 << 9,
    PL_ANNOTSENABLE_UNDERLINE     =   1 << 10,
    PL_ANNOTSENABLE_INK           =   1 << 11,
    PL_ANNOTSENABLE_FATTACH       =   1 << 12,
    PL_ANNOTSENABLE_FTEXT         =   1 << 13,
    PL_ANNOTSENABLE_POPUP         =   1 << 14,
```

PL_ANNOTSENABLE_SQUIGGLY = 1 << 15,
PL_ANNOTSENABLE_POLYLINE = 1 << 16,
PL_ANNOTSENABLE_POLYGON = 1 << 17,
PL_ANNOTSENABLE_CARET = 1 << 18,
PL_ANNOTSENABLE_SCREEN = 1 << 19,
PL_ANNOTSENABLE_PRINTER = 1 << 20,
PL_ANNOTSENABLE_TRAPNET = 1 << 21,
PL_ANNOTSENABLE_WATERMARK = 1 << 22,
PL_ANNOTSENABLE_3D = 1 << 23,
PL_ANNOTSENABLE_ALL = 0xFFFFFFFF

} PDF_TOOL_AnnotsEnableE;

戻り値：

なし

8.12.20. インポート PDF の有効アクロフォーム設定関数

PDFAPI void pl_SetAcroFormEnableOpt(hPDF ctlp,PL_AcroFormsEnableE enableAcroForms)

機能：

インポートされる PDF の有効にするアクロフォームを設定する。

引数：

ctlp ： PDF ファイルポインタ

enableAcroForms：有効にするアクロフォーム.。複数指定は論理輪で行う。

typedef enum{

PL_ACROFORMSENABLE_NONE = 0,
PL_ACROFORMSENABLE_RADIOBTN = 1 << 0,
PL_ACROFORMSENABLE_PUSHBTN = 1 << 1,
PL_ACROFORMSENABLE_CHECKBOX = 1 << 2,
PL_ACROFORMSENABLE_LISTBOX = 1 << 3,
PL_ACROFORMSENABLE_COMBOX = 1 << 4,
PL_ACROFORMSENABLE_TEXTFD = 1 << 5,
PL_ACROFORMSENABLE_SIGNATURE = 1 << 6,
PL_ACROFORMSENABLE_ALL = 0xFFFFFFFF

} PDF_TOOL_AcroFormsEnableE;

戻り値：

なし

8.13. FormXObject

FormXObject はコンテントストリームであり、ページ記述と同様の関数呼び出しで定義する。タイリングパターン同様に pl_BgnForm 関数から pl_EndForm 関数間で定義を行いインデクスを取得した後、pl_PutForm 関

数により、ページ内に描画する。

8.13.1. FormXObject 定義開始関数

PDFAPI PL_ERROR pl_BgnForm(hPDF ctlp,const PL_RectangleTD& bBox)

機能：

Form XObject の定義を開始する。

引数：

ctlp ： PDF ファイルポインタ

bBox： 矩形を定義する。通常のページ記述の用紙サイズ設定に相当する。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.13.2. FormXObject 定義終了関数

PDFAPI PL_ERROR pl_EndForm(hPDF ctlp,int& fmIdx)

機能：

Form XObject の定義を終了する。定義した FormXObject のインデクスが戻る。

引数：

ctlp ： PDF ファイルポインタ

fmIdx： FormXObject インデクスが戻る。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.13.3. FormXObject 出力関数

PDFAPI PL_ERROR pl_PutForm(hPDF ctlp,int fmIdx,float x,float y, float w,float h)

機能：

Form Xobject をカレントページに出力する。

引数：

ctlp ： PDF ファイルポインタ

fmIdx： FormXObject インデクス(pl_EndForm で取得した値)

x,y： FormXObject の左下端のカレントページ上での表示位置の座標

w,h： FormXObject を表示する領域のカレントページ上での幅と高さ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.13.4. ピースインフォ指定関数

PDFAPI PL_ERROR pl_SetPieceInfo(hPDF ctlp,const PL_PieceInfoTD& pieceInfo)

機能：

Form Xobject をカレントページに出力する。

引数：

ctlp ： PDF ファイルポインタ

pieceInfo： ピースインフォ

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_PieceInfoTD の定義うを以下に示す

```
typedef struct {
        char* lpAppName;        // アプリケーション名
        char* lpAppKey;         // アプリケーションキー
        char* lpAppValue;       // 内容
} PL_PieceInfoTD;
```

8.14. Output Intent 設定関数

PDF/X のサポートに対して、二種の方式がある。

- ❖ 標準の出力デバイスを使用場合、出力デバイスの名称を設定する。
- ❖ ColorProfile に定義される出力デバイスを使用する場合は、ColorProfile のハンドル、及び出力デバイスの属性を設定する必要がある。

二種の方式に対して異なったロード方法を用意する。ロード後、同様の方式で設定を行う。

8.14.1. 標準出力インテントの Load 関数

```
PDFAPI PL_ERROR pl_LoadStdOutputIntent(hPDF ctlp,
                        PL_OutIntentStdE stdOutputIntent,PL_OPHandle& hOutputIntent,
                        const PL_OI_PropTD* lpOutputIntentProp = NULL);
```

機能:

標準の出力インテントをロードする。

引数:

ctlp：PDF ファイルポインタ

stdOutputIntent: 標準出力インテント。定義を以下に示す。

```
typedef enum {
        //CGATS characterization data
        PL_OI_STD_CGATSTR001 = 0,
        //FOGRA Characterization data
        PL_OI_STD_FOGRA1,
        PL_OI_STD_FOGRA2,
        PL_OI_STD_FOGRA3,
```

PL_OI_STD_FOGRA4,
PL_OI_STD_FOGRA5,
PL_OI_STD_FOGRA6,
PL_OI_STD_FOGRA7,
PL_OI_STD_FOGRA8,
PL_OI_STD_FOGRA9,
PL_OI_STD_FOGRA11,
PL_OI_STD_FOGRA12,
PL_OI_STD_FOGRA13,
PL_OI_STD_FOGRA14,
PL_OI_STD_FOGRA15,
PL_OI_STD_FOGRA16,
PL_OI_STD_FOGRA17,
PL_OI_STD_FOGRA18,
PL_OI_STD_FOGRA19,
PL_OI_STD_FOGRA20,
PL_OI_STD_FOGRA21,
PL_OI_STD_FOGRA22,
PL_OI_STD_FOGRA23,
PL_OI_STD_FOGRA24,
PL_OI_STD_FOGRA25,
PL_OI_STD_FOGRA26,
PL_OI_STD_FOGRA27,
PL_OI_STD_FOGRA28,
PL_OI_STD_FOGRA29,
PL_OI_STD_FOGRA30,
PL_OI_STD_FOGRA31,
PL_OI_STD_FOGRA32,
PL_OI_STD_FOGRA33,
PL_OI_STD_FOGRA34,
PL_OI_STD_FOGRA35,
PL_OI_STD_FOGRA
PL_OI_STD_FOGRA36,
PL_OI_STD_FOGRA37,
PL_OI_STD_FOGRA38,
PL_OI_STD_FOGRA39,
PL_OI_STD_FOGRA40,
//IFRA Characterization data

PL_OI_STD_IFRA22,
PL_OI_STD_IFRA26,
PL_OI_STD_IFRA28,
PL_OI_STD_IFRA30,
//Japan Color Characterisation data
PL_OI_STD_JC200103,
PL_OI_STD_JC200104,
PL_OI_STD_JCN2002,
PL_OI_STD_JCW2003,
//System Brunner characterization data
PL_OI_STD_EUROSB104,
PL_OI_STD_EUROSB204
} PL_OutIntentStdE;

hOutputIntent:標準出力インテントのハンドルを戻す

lpOutputIntetProp：Output Intent の属性。詳しくは説明を参照

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_OI_PropTD の定義うを以下に示す

```
typedef struct {
        UCHAR_t* lpUCondition;    // (オプション) 出力インテント辞書の OutputCondition 設定値
        char* lpConditionIdentifier; // (必須) 同上。出力インテント辞書の
                                     //   OutputConditionIdentifier 設定値
        char* lpRegistryName;     // (オプション) 出力インテント辞書の RegistryName 設定値
        UCHAR_t* lpUInfo;         // (必須) 出力インテント辞書の Info 設定値
} PL_OI_PropTD;
```

8.14.2. 汎用出力インテントの Load 関数

PDFAPI PL_ERROR pl_LoadGenOutputIntent(hPDF ctlp, PL_CPHandle hColorProfile,
const PL_OI_PropTD& outputIntentProp, PL_OPHandle& hOutputIntent)

機能:

汎用のカラープロファイル付き出力インテントをロードする。

引数:

ctlp：PDF ファイルポインタ

hColorProfile:color profile のハンドル

outputIntentProp: Output Intent の属性。詳しくは説明を参照

hOutputIntent:出力インテントのハンドルを戻す

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_OI_PropTD の定義うを以下に示す

```
typedef struct {
        UCHAR_t* lpUCondition;    // (オプション) 出力インテント辞書の OutputCondition 設定値
        char* lpConditionIdentifier; // (必須) 同上。出力インテント辞書の
                                     //   OutputConditionIdentifier 設定値
        char* lpRegistryName;     // (オプション) 出力インテント辞書の  RegistryName 設定値
        UCHAR_t* lpUInfo;         // (必須) 出力インテント辞書の Info 設定値
} PL_OI_PropTD;
```

8.14.3. Output Intent 設定関数

```
PDFAPI PL_ERROR pl_SetOutputIntent(hPDF ctlp, PL_OPHandle hOutputIntent,
                                   PL_OutputIntent_OptionE opiOption=PL_OUTPUTINTENT_PDFX);
```

機能:

Output Intent を設定する。

引数:

ctlp：PDF ファイルポインタ

hOutputIntent: 設定する Output Intent のハンドル

opiOtion：出力インテントが PDF/X 用であるか、PDF/A 用であるかを設定する。定義を以下に示す。

```
typedef enum{
        PL_OUTPUTINTENT_NONE = 0,
        PL_OUTPUTINTENT_PDFX = 1 << 0,        // PDF/X 用
        PL_OUTPUTINTENT_PDFA = 1 << 1         // PDF/A 用
}PL_OutputIntent_OptionE;
```

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.15. タグ付 PDF

8.15.1. Attribute の Load 機能

さまざまな要素に対応する属性が存在するため、種類別に設定する。

① Layout Attribute の Load 機能:

Layout Attribute が BLSE、ILSE、Illustration Element に適用される。こららは別々に設定する。

- BLSE の Layout Attribute のロード

インタフェース：

```
PDFAPI PL_ERROR pl_LoadLayoutAttr_BLSE (hPDF ctlp,
```

const PL_MK_StdLayoutAttr_BlseTD& stdBlseLayAttr, PL_MK_SAHandle& hStdBlseLayAttr);

機能：

BLSE の Layout attribute をロードし、このハンドルを取得する。このハンドルは Marked Content に設定される関数 pl_bgnMKConternt()の引数として使用する。

引数：

ロードする BLSE の Layout attribute

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_ MK_STDLAYOUTATTR_BLSETD の定義を以下に示す

```
typedef struct{
        PL_BlseOptionE              blseOption;         // オプションフラグ
        PL_MK_StdLayoutAttr_GenTD   generalattr;// BLSE の一般的なプロパティ
        float     spaceBefore;      // BLSE の前の追加空白の総数
        float     spaceAfter;       // BLSE の後ろの追加空白の総数
        float     startIndent;      // BLSE の参照領域の開始端からの距離
        float     endIndent;        // BLSE の参照領域の終了端からの距離
        float     textIndent;       // BLSE の参照領域の開始端からの追加距離
        PL_AttrLayoutTextAlignE   textAlign;
                                    // BLSE の行のインライン進行方向アラインメント
        PL_RectangleTD  bBox;       // バウンディングボックス
        Float     width;            // 要素のコンテント矩形の望ましい幅
        HBOOL  bAutoWidth;          // width の設定の有無(設定がある場合は、HFALSE とする)
        Float     height;           // 要素のコンテント矩形の望ましい高さ
        HBOOL  bAutoHeight;         // height 指定の有無(設定がある場合は HFALSE とする)
        PL_AttrLayoutBlockAlignE  blockAlign;
                                    // テーブルセル内のブロック進行方向アラインメント
        PL_AttrLayoutInlineAlignE inLineAlign;
                                    // テーブルセル内のインライン進行方向アラインメント
        PL_AttrLayoutBorderStyleE       tBorderStyle;
                                    // テーブルセルの各境界線のスタイル(デフォルト  none)
        float     tPadding;         // テーブルセルのコンテント矩形と
                                    // その周囲の境界線とのオフセット(デフォルト 0)
} PL_MK_StdLayoutAttr_BlseTD;
```

この中の blseOption はデフォルト値を設定しないデータ項目に対して設定されるものである。 選択可能な項目を以下に示す。

```
typedef enum {
```

```
        PL_BO_NONE    = 0,
        PL_BO_HEIGHT = 1 << 0,
        PL_BO_WIDTH   = 1 << 1,
        PL_BO_BBOX    = 1 << 2
    } PL_BlseOptionE;
```

PL_MK_StdLayoutAttr_GenTD 構造は ILSE 共通の部分である。その定義を以下に示す。

```
    typedef struct{
        PL_GeneralOptionE       generalOption;    /          / オプションフラグ
        PL_AttrLayoutPlacementE  placement;
                // 周囲の参照領域およびその他のコンテントに対するその要素の配置
        PL_AttrLayoutWriteModeE writeMode;              // レイアウトの進行方向
        PL_ColorTD      backgroundColor;                // (PDF 1.5) 背景色
        PL_ColorTD      borderColor[4];                 // (PDF 1.5) 境界線色
        PL_AttrLayoutBorderStyleE       borderStyle[4];  // (PDF 1.5)境界線のスタイル
        float     borderThickness[4];                   // (PDF 1.5) 境界線の太さ
        float     padding[4];       // コンテント矩形の要素と周囲の境界の間のオフセット
        PL_ColorTD      color;
                // テキスト、テーブルの境界線、テキストの装飾に使用される色
    } PL_MK_StdLayoutAttr_GenTD;
```

generalOption はデフォルト値を設定しないデータ項目に対して設定されるものである。選択可能な項目を以下に示す。

```
    typedef enum {
        PL_GO_NONE                  = 0,
        PL_GO_PLACEMENT             = 1 << 0,
        PL_GO_BACKGROUNDCOLOR       = 1 << 1,
        PL_GO_BORDERCOLOR           = 1 << 2,
        PL_GO_BORDERSTYLE           = 1 << 3,
        PL_GO_BORDERTHICKNESS       = 1 << 4,
        PL_GO_PADDING               = 1 << 5,
        PL_GO_COLOR                 = 1 << 6
    } PL_GeneralOptionE;
```

- ILSE の Layout Attribute のロード

インタフェース:

```
PDFAPI PL_ERROR pl_LoadLayoutAttr_ILSE (hPDF ctlp,
                    const PL_MK_StdLayoutAttr_IlseTD& stdIlseLayAttr,
                    PL_MK_SAHandle& hStdIlseLayAttr);
```

機能：

ILSE の Layout attribute をロードし、ハンドルを取得する。このハンドルは Marked Content に設定される関数 pl_bgnMKConternt()の引数として使用する。

引数：

ロードする ILSE の Layout attribute

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_ MK_STDLAYOUTATTR_ILSETD の定義を以下に示す。

```
typedef struct {
        PL_IlseOptionE    ilseOption; //  オプションフラグ
        PL_MK_StdLayoutAttr_GenTD        generalattr; // ILSE の一般的なプロパティ
        float       lineHeight;
                // enumLineHeight が ATTR_LAYOUT_LINEHEIGHT_CUSTOM  に
                //  等しい場合に使用される
        PL_AttrLayoutLineHeightE        enumLineHeight;
                //  要素のブロック進行方向における望ましい高さ(デフォルトユーザ空間単位)
        float       baselineShift;       //  要素のベースラインの親要素に対するシフト量
                                         // (デフォルトユーザ空間単位、デフォルト値  0)
        PL_AttrLayoutTextDecorationE    textDecorationType;
                                         //  その要素のテキストに適用される装飾
        PL_ColorTD        textDecorationColor; //(PDF 1.5)  テキストの装飾の色
        unsigned int        textDecorationThickness; //(PDF 1.5)テキスト装飾の太さ
        PL_AttrLayoutRubyAlignE        rubyAlign;
                                         // (PDF 1.5)  ルビ部分愛の行のアラインメント
        PL_AttrLayoutPositionE    rubyPosition;
                                         //(PDF 1.5), ルビ内の RB 要素に対する RT 構造要素の配置
        float       glyphOrientationAngle;
                                         //(PDF 1.5)  参照領域の上端に対するグリフ上端の時計
                                         //  方向の回転角度を示す数値(90 の倍数  -180～360 の範囲)
        HBOOL  bAutoGlyphOrientation;
                //  glyphOrientationAngle の指定の有無。指定があるばあいは HFALSE とする
} PL_MK_StdLayoutAttr_IlseTD;
```

ilseOption はデフォルト値を設定しないデータ項目に対して設定するものである。選択可能な項目を以下に示す。

```
typedef enum {
        PL_IO_NONE                                  = 0 ,
        PL_IO_TEXTDECORATIONCOLOR                   = 1 << 0 ,
```

PL_IO_LINEHEIGHT = 1 << 1 ,

PL_IO_GLYPHORIENTATION = 1 << 2 ,

PL_IO_TEXTDECORATIONTHICKNESS = 1 << 3

} PL_IlseOptionE;

- Illustration Element の Layout Attribute のロード

インタフェース:

```
PDFAPI PL_ERROR pl_LoadLayoutAttr_ILLUE(hPDF ctlp,
                    const PL_MK_StdLayoutAttr_IllueTD& stdIllueLayAttr,
                    PL_MK_SAHandle& hStdIlluLayAttr)
```

機能：

Illustration Element の Layout attribute をロードしハンドルを取得する。このハンドルは Marked Content に設定される関数 pl_bgnMKConternt()の引数として使用する。

引数：

ロードする Illustration Element の Layout attribute

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_ MK_STDLAYOUTATTR_ ILLUETD の定義を以下に示す

typedef struct {

PL_MK_StdLayoutAttr_IlseTD ilse; // Illustration Element の ILSE 部分

PL_MK_StdLayoutAttr_BlseTD blse; // Illustration Element の BLSE 部分

} PL_MK_StdLayoutAttr_IllueTD;

② List Attribute のロード

インタフェース:

```
PDFAPI PL_ERROR pl_LoadListAttr (hPDF ctlp,
                    const PL_MK_StdListAttrTD& stdListAttr ,
                    PL_MK_SAHandle& hStdListAttr);
```

機能：

list attribute をロードしハンドルを取得する。このハンドルは Marked Content を設定する関数 pl_bgnMKConternt()の引数として使用する。

引数:

Load する stdListAttr の list attribute

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_ MK_ STDLISTATTRTD の定義を以下に示す

```
typedef struct {
        PL_AttrListNum listNum;     // リストの番号形式
} PL_ MK_ STDLISTATTRTD;
```

③ Table Attribute のロード

インタフェース:

PDFAPI PL_ERROR pl_LoadTblAttr(hPDF ctlp, const PL_MK_StdTblAttrTD& stdTblAttr, PL_MK_SAHandle& hStdTblAttr);

機能：

table attribute をロードし、そのハンドルを取得する。このハンドルは Marked Content を設定する関数 pl_bgnMKConternt()の引数として使用する。

引数：

ロードする stdTblAttr の table attribute

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明:

PL_ MK_ STDTBLATTRTD の定義を以下に示す

```
typedef struct {
        unsigned    int rowSpan;        // テーブルのロウスパン
        unsigned    int colSpan;        // テーブルのカラムスパン
        PL_AttrTblScopeE scope;         // ヘッダセルを、それを含むその行の残りのセルに
                                        // 適用するか、それを含むカラムに適用するか、
                                        // あるいは双方に適用するかを示す
} PL_MK_StdTblAttrTD;
```

8.15.2. Marked Content の開始設定機能

PDFAPI PL_ERROR pl_BgnMKContent(hPDF ctlp, const PL_MK_StdElemTD& stdElem, const PL_MK_SAHandleArrayTD& hStdAttrHandleArray, PL_MKHandle& hMarkedContent)

機能：

Marked Content の開始を指定する。

引数:

stdElem：Marked Content を指定する。

定義を以下に示す

```
typedef struct{
        PL_MK_StdElemTypeE      type;       // StructElement 型を指定する。
        PL_MKHandle     hParent; // StructElement の親ノードを指定する。
```

```
                // デフォルト値は 0 であり、これは前要素が本要素の親ノードであることを示す。
        HBOOL           bLeaf;    // StructElement に子ノードがなければ HTRUE とする
                                  // 注釈に StructElement を指定する場合、HFALSE とする。
        UCHAR_t*        title; // StructElement の title の描画文字列を指すポインタ
        UCHAR_t*        alternate; // StructElement の代替文字列を指すポインタ
        UCHAR_t*        expanded; // StructElement の省略文字列を指すポインタ
        UCHAR_t*        actualtext; // StructElement の実文字列を指すポインタ
        char*           lang;     // "en"のようなこのテキスト
        PL_MK_ArtifactElemTD artifact; // Artifact を描画する。実際には、Artifact は
                                  //StructElement ではないため pseudo structelement と呼ぶ。
        char*           tagname;  // タグ名
        } PL_MK_StdElemTD;
```

PL_MK_ArtifactElemTD の定義を以下のように示す。

```
typedef struct
{
        PL_ArtifactOptionE      option;         // オプションフラグ
        PL_ArtifactTypeE        type;           // オプション Artifact のタイプ
        PL_RectangleTD          bbox;           // オプション バウンディングボックス
        PL_ArtifactAttachedTypeE    attached;// オプション ページネーション要素のみ
}PL_MK_ArtifactElemTD;
```

PL_ArtifactOptionE の定義を以下のように示す。

```
typedef enum {
        PL_AO_NONE              = 0 ,
        PL_AO_TYPE              = 1 << 0 ,
        PL_AO_BBOX              = 1 << 1 ,
        PL_AO_ATTACHED          = 1 << 2
} PL_ArtifactOptionE;
```

type;は Artifact の型を指定する。

```
typedef enum{
        PL_ARTIFACT_PAGINATION = 0,
        PL_ARTIFACT_LAYOUT,
        PL_ARTIFACT_PAGE,
}PL_ArtifactTypeE;
typedef enum {
        PL_ARTIFACT_ATTACHED_TOP            = 1,
        PL_ARTIFACT_ATTACHED_BOTTOM         = 2,
        PL_ARTIFACT_ATTACHED_LEFT           = 4,
        PL_ARTIFACT_ATTACHED_RIGHT          = 8
```

} PL_ArtifactAttachedTypeE;

stdAttrHandleArray は:Marked Content 使用の属性を指定する。定義を以下のようにする。

```
typedef struct{
    PL_MK_SAHandle*     pSAHArray;      // この構造要素の属性の配列
    Int                 nCount;         // この配列の要素数
}PL_MK_SAHandleArrayTD;
```

hMarkedContent: 取得した Marked Content の Handle。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

注: この関数は pl_CreatePage()と pl_ClosePage()の間にいつでも使用可能、多数の Marked Content の間にまたがるような設定はできない。ネストは可能である。

8.15.3. Marked Content の終了指定機能

PDFAPI PL_ERROR pl_endMKContent (hPDF ctlp, PL_MKHandle hMarkedContent);

機能：

Marked Content の終了を設定する。

引数:

hMarkedContent:カレント Marked Content の handle。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.15.4. Marked Content リーフの開始設定機能

PDFAPI PL_ERROR pl_BgnMKContentLeaf(hPDF ctlp, PL_MKHandle hMarkedContent);

機能：

Marked Content リーフの開始を設定する。

引数：

hMarkedContent: 取得した Marked Content の Handle。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.15.5. Marked Content 子節点の終了設定機能

PDFAPI PL_ERROR pl_EndMKContentLeaf(hPDF ctlp, PL_MKHandle hMarkedContent);

機能：

Marked Content リーフの終了を設定する。

引数：

hMarkedContent:カレント Marked Content の handle。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.15.6. Marked Content のアクティブにする設定機能

PDFAPI PL_ERROR pl_ActivateMKElement(hPDF ctlp, PL_MKHandle hMarkedContent);

機能：

Marked Content をアクティブ設定する。

引数:

hMarkedContent：マーク付きコンテントのハンドル。pl_BgnMKConte に対応し、閉じられる前でなければならない。Pseudo およびインラインレベル項目はアクティブにできない。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.16. アクション作成関数

個々で定義されるアクションは Outline、Annotaion、AcroForm などで使用される。下記が指定できる。
GoTo、GoToR、Launch、Thread、URI、 Hide、 Named、SubmitForm、ResetForm、ImportData、JavaScript
ここで示す関数は一つのアクションのハンドルを作成することができ、他の関数(例えば、AcroForm 関数)で使用することができる。
Sound、Movie Action は現在 未サポートである。

アクションに、使用する共通の構造体定義を以下に示す。

- PL_FormatTD

  定義：

```
typedef struct {
    PL_FormatDataTD      formatData;
    PL_FormatCategoryE   formatType;
} PL_FormatTD;
```

  説明：

  - formatData:　Format 情報を設定する。PL_FormatDataTD 参照
  - formatType:　Format 型。PL_EformatCategoryE 参照

- PL_Formatcategorye

  定義：

```
typedef enum{
    PL_FC_None=0,              // フォーマットなし
    PL_FC_Number,              // 数値フォーマット
    PL_FC_Percentage,          // パーセントォーマット
```

```
    PL_FC_Date,                 // 日付フォーマット
    PL_FC_Time,                 // タイムフォーマット
    PL_FC_Special               // 特殊フォーマット
} PL_FormatCategoryE;
```

❖ PL_FormatDataTD

定義：

```
typedef struct {
        union{
                PL_FM_DateFormatTypeE dateFormatType;// 日付フォーマット情報
                PL_FM_TimeFormatTypeE timeFormatType;// タイムフォーマット情報
                PL_FM_SepicalFormatTypeE specialFormatType;// 特殊フォーマット情報
        };
        PL_FM_SeperatorStyleE seperatorStyle; // 数字セパレータの型。数値フォーマット用
        int nDec;    // 小数点以下の桁数。数値フォーマット、パーセントフォーマット用
        PL_FM_NegativeNumberStyleE negStyle;// 数値のマイナスの型。数値フォーマット用
                                        //PL_FM_NegativeNumberStyle 参照
        PL_CurrencyE strCurrency; // 通貨マーク型。数字フォーマット用
        HBOOL bCurrencyPrepend; // 通貨マークを接頭語とする／しない。数値フォーマット用
} PL_FormatDataTD;
```

❖ PL_FM_SepicalFormatTypeE

定義：

```
typedef enum {
        PL_FM_ZipCode = 0,          // 郵便番号フォーマット
        PL_FM_ZipCode4,             // 郵便番号の 4 枠付きのフォーマット
        PL_FM_PhoneNumber,          // 電話番号フォーマット
        PL_FM_SocialSecurityNumber  // 社会保険番号フォーマット
} PL_FM_SepicalFormatTypeE;
```

❖ PL_FM_TimeFormatTypeE

定義：

```
typedef enum {
        PL_FM_24HR_MM = 0 ,         // 24 時制フォーマット
        PL_FM_12HR_MM,              // 12 時制フォーマット
        PL_FM_24HR_MM_SS,           // 24 時制フォーマット
        PL_FM_12HR_MM_SS            // 12 時制フォーマット
} PL_FM_TimeFormatTypeE;
```

❖ PL_FM_DateFormatTypeE

定義：

```
typedef enum {
                PL_FM_MSD=0,                    // m/d
                PL_FM_MSDSYY,                   // m/d/yy
                PL_FM_MSDSYYYY,                 // m/d/yyyy
                PL_FM_MMSDDSYY,                 // mm/dd/yy
                PL_FM_MMSDDSYYYY,               // mm/dd/yyyy
                PL_FM_MMSYY                     // mm/yy
                PL_FM_MMSYYYY,                  // mm/yyyy
                PL_FM_D_MMM,                    // d-mmm
                PL_FM_D_MMM_YY,                 // d-mmm-yy
                PL_FM_D_MMM_YYYY,               // d-mmm-yyyy
                PL_FM_DD_MMM_YY,                // dd-mmm-yy
                PL_FM_DD_MMM_YYYY,              // dd-mmm-yyyy
                PL_FM_YY_MM_DD,                 // yy-mm-dd
                PL_FM_YYYY_MM_DD,               // yyyy-mm-dd
                PL_FM_MMM_YY,                   // mmm-yy
                PL_FM_MMM_YYYY,                 // mmm-yyyy
                PL_FM_MMMM_YY,                  // mmmm-yy
                PL_FM_MMMM_YYYY,                // mmmm-yyyy
                PL_FM_MMMSpaceDDotYYYY,         // mmm d, yyyy
                PL_FM_MMMMSpaceDDotYYYY,        // mmmm d, yyyy
                PL_FM_MSDSYYSpaceHColonMMSpaceTT,           // m/d/yy h:MM tt
                PL_FM_MSDSYYYYSpaceHColonMMSpaceTT,// m/d/yyyy h:MM tt
                PL_FM_MSDSYYSpaceHHColonMM,         // m/d/yy HH:MM
                PL_FM_MSDSYYYYSpaceHHColonMM        // m/d/yyyy HH:MM
}PL_FM_DateFormatTypeE;
```

❖ PL_FM_SeperatorStyleE

定義：

```
typedef enum{
        PL_FM_SeparatorNormal=0,        // 桁区切り（,）、小数点（.）1,234.56
        PL_FM_NotSeparatorNormal,       // 桁区切りなし、小数点（.）1234.56
        PL_FM_SeparatorByDots,          // 桁区切り（.）、小数点（,）1.234,56
        PL_FM_NotSeparatorByDots        // 桁区切りなし、小数点（,）1234,56
}PL_FM_SeperatorStyleE;
```

❖ PL_FM_NegativeNumberStyleE

定義：

```
typedef enum{
        PL_FM_MinusWithNegative=0,              // マイナス記号で負数を表示する。
        PL_FM_RedColorWithNegative,             // 赤色（赤字）で負数を表示する。
        PL_FM_ParenthesisWithNegative,          // 括弧で負数を表示する。
        PL_FM_RedColorParenthesisWithNegative   // 括弧で負数を示し、かつ、数字を赤に
}PL_FM_NegativeNumberStyleE;
```

注：現在 Acrobat が 2,4 設定に対応していないと思われる

❖ PL_CurrencyE

定義：

```
typedef enum{
        PL_CR_None=0,       // 通貨マークなし
        PL_CR_Dollar,       // ドル
        PL_CR_Deutchmark,   // ドイツマーク
        PL_CR_Euro,         // ユーロ
        PL_CR_Franc,        // フラン
        PL_CR_Guilder,      // 通貨型
        PL_CR_Krona,        // クローナー
        PL_CR_Lira,         // リラ
        PL_CR_Peseta,       // ペセタ
        PL_CR_Pound,        // ポンド
        PL_CR_Yen           // 円
}PL_CurrencyE;
```

❖ PL_EventE

定義：

```
typedef enum {
        PL_EV_MouseUp=0,    // マウスアップ動作
        PL_EV_MouseDown,    // マウスダウン動作
        PL_EV_MouseEnter,   // マウスエンター動作
        PL_EV_MouseExit,    // マウスエクジット動作
        PL_EV_OnFocus,  //（あるフィールド）フォーカスを取得するイベント。
        PL_EV_OnBlur,   //（あるフィールド）フォーカスを失うイベント。
        PL_EV_Keystroke,    // キーボードを押すイベント。
        PL_EV_Format,       // フォーマットイベント
        PL_EV_Validate,     // データ確認
```

```
        PL_EV_Caculate,          // イベント総数
        PL_EV_All
    }PL_Event;
```

❖ PL_ButtonStyleE

定義：

```
typedef enum{
        PL_BTNSTYLE_QUADRIATERAL=0,     /  四角形ボタン  (■)
        PL_BTNSTYLE_CHECKMARK,          // チェックボタン  (レ)
        PL_BTNSTYLE_CIRCLE,             // 丸型ボタン      (●)
        PL_BTNSTYLE_RHOMBUS,            // 菱形ボタン      (◆)
        PL_BTNSTYLE_CROSS,              // バツ型ボタン    (×)
        PL_BTNSTYLE_STAR,               // 星型ボタン      (★)
        PL_BTNSTYLE_NUM                 // 数字型ボタン
    }PL_ButtonStyleE;
```

❖ PL_ItemTD

定義：

```
    typedef struct {
            UCHAR_t* lpIName;    // 選択（Item）の表示名称。
            UCHAR_t* lpIValue;   // 選択に示される値。
            HBOOL bSelect;       // ハイライトで選択内容を表示するかどうか
                                 // Default value:HFALSE.
    } PL_ItemTD;
```

❖ PL_ItemsTD

定義：

```
typedef struct {
            PL_ItemTD* lpItems;  // 選択情報配列（item）ポインタ。
            int iNum;            // 選択情報配列の長さ。
    } PL_ItemsTD;
```

8.16.1. pl_Action_CreateUri

PDFAPI PL_ERROR pl_Action_CreateUri(hPDF ctlp,const PL_PL_ActionUriTD& actionURI, PL_ActionHandle& hAction);

機能：

URI 動作応答を作成する。

引数：

ctlp：PDF ハンドル

actionURI：URI アクションの情報

hAction：アクションのハンドル。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_ActionUriTD の定義を以下に示す

```
typedef struct {
    UCHAR_t* lpUri ;          // アドレス情報。
    HBOOL bMap;               // マウス追跡用マーク（普通は False)。
} PL_ActionUriTD;
```

8.16.2. pl_Action_CreateGoto

PDFAPI PL_ERROR pl_Action_CreateGoto(hPDF ctlp, const PL_ActionGotoTD& actionGoto, PL_ActionHandle& hAction);

機能：

GoTo 動作応答を作成する。

引数：

ctlp： PDF ハンドル

actionGoto: Goto 動作の目的情報,詳見説明

hAction：アクションのハンドル。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_ActionGotoTD の定義を以下に示す:

```
typedef struct {
    UCHAR_t* lpDestName;// 対象ファイル名称
} PL_ActionGotoTD;
```

8.16.3. pl_Action_CreateGotoR

PDFAPI PL_ERROR pl_Action_CreateGotoR(hPDF ctlp,const PL_ActionGotoRTD& actionGotoR, PL_ActionHandle& hAction);

機能：

GoToR 動作応答を作成する。

引数：

ctlp： PDF ハンドル

actionGotoR: GotoR 動作の目的情報,詳しくは説明を参照詳見説明

hAction：アクションのハンドル。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_ActionGotoRTD の定義を以下に示す

```
typedef struct {
    UCHAR_t* lpDestFile;      // 対象ファイルの位置
    UCHAR_t* lpDestName;      // 対象ファイルの名称
    HBOOL bNewWindow;         // 新しいウィンドウで対象ファイルを開くかどうか
} PL_ActionGotoRTD;
```

8.16.4. pl_Action_CreateNamed

PDFAPI PL_ERROR pl_Action_CreateNamed(hPDF ctlp, const PL_ActionNamedTD& actionNamed, PL_ActionHandle& hAction);

機能：

(Execute Menu Item action) 実行メニューの動作項目を作成する。

引数：

ctlp：PDF ハンドル

actionNamed：実行するメニュー項目名称,

hAction：アクションのハンドル。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明:

PL_ActionNamedTD の定義を以下に示す

```
typedef struct {
    UCHAR_t* lpName;          //メニュー名
} PL_ActionNamedTD;
```

8.16.5. pl_Action_CreateJavaScript

PDFAPI PL_ERROR pl_Action_CreateJavaScript(hPDF ctlp, const PL_ActionJspTD& actionJSP,PL_ActionHandle& hAction);

機能：

JavaScript 応答を作成する。

引数：

ctlp：PDF handle

actionJSP：実行する JavaScript 内容

hAction：アクションのハンドル。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明:

PL_ActionJspTD 定義如下:

```
typedef struct {
    UCHAR_t* lpJSContent;     // java script コード
} PL_ActionJspTD;
```

8.16.6. pl_Action_CreateImpData

PDFAPI PL_ERROR pl_Action_CreateImpData(hPDF ctlp, const PL_ActionImpDataTD& actionImpData,
PL_ActionHandle& hAction);

機能：

データ導入の応答を作成する。

引数：

ctlp：PDF handle

actionImpData：入力するデータ内容

hAction：アクションのハンドル。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明:

PL_ActionImpDataTD の定義を以下に示す

```
typedef struct {
    UCHAR_t* lpFDFFileName;          // FDF ファイル名
} PL_ActionImpDataTD;
```

8.16.7. pl_Action_CreateLaunch

PDFAPI PL_ERROR pl_Action_CreateLaunch(hPDF ctlp, const PL_ActionLaunchTD& actionLaunch,
PL_ActionHandle& hAction);

機能：

ファイルを開く応答を作成する。

引数：

ctlp：PDF handle

actionLaunch：ファイルを開く情報,詳見説明

hAction：アクションのハンドル。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明:

PL_ActionLaunchTD の定義を以下に示す

```
typedef struct {
```

UCHAR_t* lpLaunchName; //開くファイル名

HBOOL bNewWindow;　//新しいウィンドウで対象ファイルを開くかどうか

} PL_ActionLaunchTD;

8.16.8. pl_Action_CreateSHField

PDFAPI PL_ERROR pl_Action_CreateSHField(hPDF ctlp, const PL_ActionSHFieldTD& actionSHField, PL_ActionHandle& hAction);

機能：

Form を隠す応答を作成する。

引数：

ctlp：PDF handle

actionSHField：隠す Form の情報,詳見説明

hAction：アクションのハンドル。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明:

PL_ActionSHFieldTD の定義を以下に示す

typedef struct {

UCHAR_t* lpFieldText;　　//フォーム名

HBOOL bSH;　　//隠す／隠さない

} PL_ActionSHFieldTD;

8.16.9. pl_Action_CreateSubmitForm

PDFAPI PL_ERROR pl_Action_CreateSubmitForm(hPDF ctlp, const PL_ActionSubmitFormTD& actionSubmitForm,, PL_ActionHandle& hAction);

機能：

Form を提出する応答を作成する。

引数：

ctlp：PDF handle

actionSubmitForm：提出する Form の情報,詳見説明

hAction：アクションのハンドル。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明:

PL_ActionSubmitFormTD の定義を以下に示す

typedef struct {

UCHAR_t* lpUri;　　//アドレス

UCHAR_t** lpFieldsArray; //フォーム名配列

int　iFieldsNum;　　//フォーム名数量

int　iFlags;　　// submit-form action のマーク。PDF1.4 参考ドキュメントの TABLE 8.62 Flags

//for submit-form actions をご覧ください。

} PL_ActionSubmitFormTD;

8.16.10.　　pl_Action_CreateResetForm

PDFAPI PL_ERROR pl_Action_CreateResetForm(hPDF ctlp, const PL_ActionResetFormTD& actionResetForm,

PL_ActionHandle& hAction);

機能：

Form を改めて設定する応答を作成する。

引数：

ctlp：PDF handle

actionResetForm：改めて設定する Form の情報,詳見説明

hAction：アクションのハンドル。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明:

PL_ActionResetFormTD の定義を以下に示す

typedef struct {

UCHAR_t** lpFieldsArray; //フォーム名配列

int　iFieldsNum;　　//フォーム名数量

int　iFlags;　　// submit-form action のマーク。PDF1.4 参考ドキュメントの TABLE 8.62 Flags

/ for submit-form actions をご覧ください。

} PL_ActionResetFormTD;

8.16.11.　　pl_Action_CreateThread

PDFAPI PL_ERROR pl_Action_CreateThread(hPDF ctlp, const PL_ActionThreadTD& actionThread,

PL_ActionHandle& hAction);

機能：

文章内容を読取る（read Article）応答を作成する。

引数：

ctlp：PDF handle

actionThread：読取る文章内容（read Article）情報,詳しくは説明を参照

hAction：アクションのハンドル。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明:

PL_ActionThreadTD の定義を以下に示す

```
typedef struct {
UCHAR_t* lpThreadName;        //スレッド名
    int   iDestiThread;          //スレッド番号
    int   iBeadDestiThread;  //ビードスレッド番号
} PL_ActionThreadTD;
```

8.16.12. pl_Action_AddAction

PDFAPI PL_ERROR pl_Action_AddAction(hPDF ctlp,PL_ActionHandle hAction,PL_ActionHandle hActionParent,
PL_ActionHandle hActionTree);

機能：

Action ツリーに新しい結節点を追加する。

引数：

ctlp：PDF handle

hAction：目的ツリーのハンドル番号

hActionParent:ファーザ・ノードハンドル番号。

hActionTree:挿入される子ノード（或はツリー）ハンドル番号。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.17. 対話フォームオブジェクト出力関数

対話フォーム（AcroForm）は TextField,Radio Button,Push Button,Check Box,Choice Field などのオブジェクトを PDF 文書のページ上の場所に関連付ける、ページのコンテンツとは独立したオブジェクトである。

各対話フォームについての詳細は PDF 仕様書参照。

すべての対話フォームにおいて、下記の手順で設定する必要がある。

- まず pl_AcroForm_LoadXXX 関数を呼び出して対話フォームを Load する。
- pl_AcroForm_Set を関数を呼び出して対話フォームを設定する。

各対話フォーム設定で使用される共通の構造体の定義を以下に示す。

```
typedef struct {
```

PL_WCommonFlagE wCommFlag; //(Optional) Flags specifying which parameters will be used.

PL_RectangleTD Rect;//(Required)The location of the annotation.

PL_ColorTD ColorBC;//(Optional)The color used for the border.The valid color:

//PL_CS_NONE=0,//no color,transparent

//PL_CS_DEVICERGB,PL_CS_DEVICEGRAY,PL_CS_DEVICECMYK,//Device color space

PL_ColorTD ColorBG;//(Optional)The color used for the background.

//The valid color same as described above for ColorBC.

UCHAR_t* T; //(Optional)The partial field name.

UCHAR_t* TU; //(Optional)An alternate field name, to be used in place of the actual

//field name wherever the field must be identified in the user interface

//(such as in error or status messages referring to the field).

//This text is also useful when extracting the document's contents in

//support of accessibility to disabled users or for other purposes.

UCHAR_t* TM; //(Optional)The mapping name to be used when exporting interactive

//form field data from the document.

PL_HighlightMode H; //(Optional) The annotation □ s highlighting mode, the visual effect to be

//used when the mouse button is pressed or held down inside its active area

PL_BorderTD BS; //(Optional) This key overrides the Border key. The value of this key is a

//dictionary that specifies several attributes related to the border of the annotation.

HBOOL bReadOnly; //(Optional) If set, the user may not change the value of the field.

//Any associated widget annotations will not interact with the user; that is, they

//will not respond to mouse clicks or change their appearance in response to

//mouse motions. This flag is useful for fields whose values are computed or

//imported from a database.

HBOOL bRequired; //(Optional) If set, the field must have a value at the time it is exported by a

//submit-form action

HBOOL bNoExport; //(Optional) If set, the field must not be exported by a submit-form action

int F; //(Optional)Annotation flag,a set of flags specifying various characteristics of the

//annotation. Default value: 0.

UCHAR_t* lpValue; //(Optional) The field's value.

int rDegrees; //(Optional) The number of degrees by which the widget annotation

//is rotated counterclockwise relative to the page. The value must

//be a multiple of 90.Default value: 0.

PL_FormatTD eFormat; //(Optional) set format action

PL_ActionHandle hActionTree[PL_EV_All]; //(Optional)

} PL_WCommDataTD;

上記の各項目について

- wCommFlags--ドキュメント名
- Rect--フォーム座標。
- ColorBC--フォーム枠色
- ColorBG--フォーム背景色。
- T--フォーム部分名。
- TU--フォーム部分名。
- TM--フォーム部分名。
- H--フォーム高亮度表示。
  - PL_HL_INVERT(埋め込み)
  - PL_HL_NONE(普通).。
  - PL_HL_OUTLINE(アウトライン)
  - PL_HL_PUSH(押し)。
- BS--フォーム枠データは BorderTD 参照
- BreadOnly--フォームは読み取り専用か否か
- Brequired--フォームは submit-form に導き出される否か
- BnoExport--フォームはきっと submit-form に導き出されるか否か
- F--フォーム表示設定
- LpValue--フォーム名
- rDegrees--フォーム回転度数
- eFormat--フォームフォーマットは EformatTD 参照
- hActionTree--フォーム Action ハンドル。

8.17.1. pl_AcroForm_LoadTextField

PDFAPI PL_ERROR pl_AcroForm_LoadTextField(hPDF ctlp,const PL_WCommDataTD& wCommData, const PL_TextFieldTD& textField,PL_AcroFormHandle& hAcroForm);

機能：

一つの Text Acroform を構築する。

引数：

ctlp：PDF handle

wCommData：Acroform の共通情報,前の定義を参照

textField：テキスト AcroForm 内容情報.。説明を参照

hAcroForm: AcroForm のハンドル番号。

返し値：

成功なら、0 が返す、でないと、エラーコードが返す(エラーコード番号を参照してください)。

説明：

PL_TextFieldTD の定義は以下通り:

```
typedef struct {
    PL_FT_AlignmentE    alignment; //テキスト揃え方式。.
    PL_FontTD textFont;             //テキストフォント。
    nt defFontNum;                  //テキストデフォルトフォント数
    PL_FontTD defFont[5];           //テキストデフォルトフォント
   int maxLen;                      //テキスト最大限長さ
```

```
    HBOOL bMultiline;              //多行である／でない
    HBOOL bPassword;               //暗号化する／しない
   HBOOL bFileselect;              // submit 選択ファイルが有効であるかどうか
  HBOOL bDonotspellcheck;          //スペリングチェックのプログラムを使用するかどうか
   HBOOL bDonotscroll;             //スクロールがある／ない
} PL_TextFieldTD;
```

8.17.2. pl_AcroForm_LoadRadioCheckButton

```
PDFAPI PL_ERROR pl_AcroForm_LoadRadioCheckButton(hPDF ctlp,
        const PL_WCommDataTD& wCommData,
        const PL_RadioChkBtnTD& radioChkButton,PL_AcroFormHandle& hAcroForm);
```

機能：

一つの Radio Check Button Acroform を構築する。

引数：

ctlp：PDF handle

wCommData：Acroform の共通情報,。前の定義を参照

radioChkButton：Radio Check Button AcroForm 内容情報.は説明を参照

hAcroForm: AcroForm のハンドル番号。

返し値：

成功なら、0 が返す、でないと、エラーコードが返す(エラーコード番号を参照してください)。

説明：

PL_RadioChkBtnTD 定義如下:

```
typedef struct {
    HBOOL bRadioBtn;               // RadioButton 或は CheckBox
    PL_ButtonStyle btnStyle;       //バートン型
    float fSize;                   //フォンートサイズ
    PL_ColorTD fColor;             //フォンート色
    HBOOL bDefStateOn;             //デフォルトの表示アイコンを使用するかどうか
}PL_RadioChkBtnTD;  //radio and check button
```

8.17.3. pl_AcroForm_LoadPushButton

```
PDFAPI PL_ERROR pl_AcroForm_LoadPushButton(hPDF ctlp,
        const PL_WCommDataTD& wCommData,
        const PL_PushBtnTD& pushButton,PL_AcroFormHandle& hAcroForm);
```

機能：

一つの Push Button Acroform を構築する。

引数：

ctlp：PDF handle

wCommData：Acroform の共通情報,請参見前面的定義

pushButton：Push Button AcroForm 内容情報.見説明

hAcroForm：作成した AcroForm のハンドル番号。

返し値：

成功なら、0 が返す、出ないと、エラーコードが返す（エラーコード番号を参照してください）。

説明：

PL_PushBtnTD の定義は以下通り:

```
typedef struct {
    int rotateDegrees; //回転角度
    PL_PushBtnLayoutTD normalLayout;/正常に表示したデータ。
    PL_PushBtnLayoutTD rolloverLayout;//転がって表示したデータ。
    PL_PushBtnLayoutTD downLayout;  //クリックして表示したデータ。
    PL_IconFitTD iconFit;           //表示アイコン。
    PL_IconTextPos iconTextPos;     //アイコン座標。
    PL_FontTD textFont;             //テキストフォント
}PL_PushBtnTD
```

8.17.4. pl_AcroForm_LoadChoiceField

```
PDFAPI PL_ERROR pl_AcroForm_LoadChoiceField(hPDF ctlp,
        const PL_WCommDataTD& wCommData,
        const PL_ChoiceFieldTD& choiceField,PL_AcroFormHandle& hAcroForm);
```

機能：

一つの Choice Acroform を構築する。

引数：

ctlp：PDF handle

wCommData：Acroform の共通情報。前の定義を参照

choiceField：Choice AcroForm 内容情報.説明を参照

hAcroForm：作成した AcroForm のハンドル番号。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

説明：

PL_ChoiceFieldTD の定義は以下通り：

```
typedef struct {
    PL_FontTD textFont;          //テキストフォント。
    HBOOL bComboBox;// ComoboBox 或は ListBox
    HBOOL bEditable;      //編集可能がある／ない
    HBOOL bSorted;          // ICheck--スペリング
    HBOOL bMultiSelect;   //多選択である／でない
    HBOOL bNotSpellCheck;   //スペリングチェックのプログラムを使用するかどうか
    HBOOL bCommitOnSelChange;      //(選ぶ時に、表示を最新する／しない
    PL_ItemsTD items;      //フォームに表示するデータ。
} PL_ChoiceFieldTD;
```

8.17.5. pl_AcroForm_Set

PDFAPI PL_ERROR pl_AcroForm_Set(hPDF ctlp,PL_AcroFormHandle hAcroForm,
PL_AcroFormHandle hAcroFormParent=0,int pgNo=0);

機能：

指定される Acroform を指定頁に置く。

引数：

ctlp：PDF handle

hAcroForm：設定する Acroform　Handle 番号

hAcroFormParent：カレント Acroform のファーザノードのハンドル番号。

pgNo:Acroform の出力頁番号。

戻り値：

正常終了の場合、0 を戻す。それ以外の場合、エラーコードを戻す

8.18. エラー情報取得関数

本ライブラリがエラーを戻した場合のエラーメッセージなどを取得するために使用する関数である。

8.18.1. エラーメッセージ取得関数

PDFAPI wchar_t* pl_GetErrorMessage(hPDF ctlp)

機能：

発生したエラーのメッセージを取得する。

引数：

ctlp ： PDF ファイルポインタ

戻り値：

エラーメッセージ文字列

9. フォント設定

9.1. サポートされるフォント形式

本ライブラリでサポートされるフォントは以下のフォント形式である。

- Adobe Type1 フォント
- TrueType フォント
- OpenType フォント

各フォント形式に関する使用上の注意事項については後述する。

9.2. フォント設定ファイル

本ライブラリを使用して文字出力を行う場合、使用するフォントを格納したフォルダなどに関する情報をフォント設定ファイルに記述して指定する必要がある。このフォント設定ファイルについて説明する。

9.2.1. 概要

フォント設定ファイルは簡単な XML ファイルである。DTD は添付の font-config.dtd である。このフォント設定ファイルのルート要素は属性無しの <font-config> であり、<font-config> の子要素は <name-processing-mode>、<font-folder>の 2 種類である。

<name-processing-mode>は 1 つだけ指定可能であり、Type1 フォントの名称の指定方法を定義するものである。指定する場合は必ず<font-folder>より前に記述する。

<font-folder>は複数記述することが可能であり、フォントファイルが格納されているフォルダの指定と、そのフォルダ内に格納されているフォントに関する情報を記述する。

基本的には ここに指定されたフォルダ内に格納されているフォントが自動的に検索され､そのまま使用することができる。

なお、Windows 環境では、フォント設定ファイルが存在しない場合、下記の name-processing-mode で WindowsName モードが指定され、フォントフォルダには Windows のフォントフォルダが指定された状態で動作する。このため、Windows で通常使用される TrueType フォント、TrueType アウトラインを持つ OpenType フォントを、Windows にインストールされた状態で使用する場合、フォント設定ファイルは不要である。

9.2.2. フォント設定ファイルの要素・属性

フォント設定ファイルのルート要素 <font-config>配下の要素と属性について記述する。

要素名	位置	説明
name-processing-mode	font-config 下 (複数指定不可)	属性として”mode”を持つ。 Type 1 フォントのフォント名を、Windows の表示名を用いて指定するか否かを定義する。 "mode" 属性に、"default" または "windows-name" を指定する。 既定値は "default" であり、この場合、FamilyName による処理となる。

要素名	位置	説明
		Windows の表示名で指定を行う場合、 "windows-name"を指定する。 例：<name-processing-mode mode= "windows-name"/>
font-folder	font-config 下 (複数指定可)	属性として”path”を持つ。 "path" 属性に、フォントフォルダの位置を指定する。 複数のフォルダに格納されているフォントを使用する場合、それにあわせて、この要素を複数指定する。 例：<font-folder path="c:¥FontsFolder "> .... </font-folder> このフォルダ内に格納されているフォントは自動的に検索され、使用可能となる。追加の情報がある場合は、この要素の下に記述する。
glyph-list	font-folder 下 (複数指定可、 空要素)	属性として “file”と”afm”をもつ。 “file”で指定されるフォントファイルに対して、使用するグリフリストのファイル名を”afm”に指定する。 “file”には Type1 フォントの.AFM ファイル名を指定する。 グリフリストを使用するフォントがフォルダ内に複数ある場合、それぞれについてこの要素を使用して定義する。 例：<glyph-list file="ZapfDingbats.txt" afm="ZD_____.AFM"/>
skip-glyphname-mapping	font-folder 下 (複数指定可、 空要素)	属性として”afm”を持つ。 Type 1 フォントの、Unicode とグリフ名の対応付けをスキップする場合、そのフォントの.AFM ファイル名を”afm”属性に指定する。 スキップするフォントがフォルダ内に複数ある場合、それぞれについてこの要素を使用して記述する。 例：<skip-glyphname-mapping afm="CR_____.AFM"/>
font-exclude	font-folder 下 (複数指定可、 空要素)	属性として”file”を持つ。 フォルダ内に検索対象(処理対象)から除外するフォントが存在する場合、そのファイル名を”file”属性で指定する。Type1 フォントの場合、拡張子に.AFM または.PFM を持つファイルを指定する。
font-alias	font-folder 下 (複数指定可)	属性として”file”と”entry”を持つ。 フォントファイルに定義されているフォントファミリー名とは別の名称で上位からフォント指定を行う場合に、その別名を子要素に定義する。 対象となるファイル名を”file”属性に指定する。 Type1 フォントでは拡張子に.AFM または.PFM を持つファイルを指定する。 拡張子が.TTC のファイルは、一つのファイル内に複数のフォントが格納されている。”entry”属性には、この中の別名を定義するフォントの番号を指定する(1 オリジン、省略値は 1)。
alias	font-alias 下 (複数指定可、 空要素)	属性として”family-name”、”weight”、”italic”を持つ。 フォントファイルに与える別名を”faily-name”、ウェイト値を”weight”、斜体か否かの情報を”italic”に、それぞれ記述する。

要素名	位置	説明
		別名を定義した場合、フォント内に定義される FamilyName に優先される。 他の別名と重複となるような定義が検出された場合、本ライブラリではエラー(PL_ERR_F_Alias)が発生する。 “weight”属性では、文字の太さを指定する。 “100”(細い)から”900”(太い)の間の 100 単位の数値、または “normal”、”bold” を指定する。 “normal”は”400”、”bold”は”700”に相当する。省略時はフォントファイルの定義値が使用される。 “italic”属性には、斜体の場合 “HTRUE”、そうでない場合は “HFALSE”を指定する。省略時はフォントファイルの定義値が使用される。

9.3. Type1 フォント

9.3.1. フォントのファイル構成

Type1 フォントは以下のファイルから構成される。

拡張子	説明
.PFB(Priter Font Binary)	バイナリ圧縮されたアウトラインデータ
.AFM(Adobe Font Metrics)	フォントの一般的な情報と、フォントメトリクス情報を含むテキストファイル。UNIX では、主に、.AFM+.PFB の組み合わせで使用される。
.PFM(Printer Font Metrics)	フォントの一般的な情報と、フォントメトリクス情報を含むバイナリファイル。Windows 上での表示名を持つ。Windows では、主に、.PFM+.PFB の組み合わせで使用される。

本ライブラリでは.AFM と.PFB、.PFM と．PFB の双方の組み合わせをサポートする。また、WindowsName モードを使用する場合、.PFM ファイルが必須となる。

その他、補足

- .PFA(Printer Font Ascii) という拡張子を持つアウトラインファイルが存在するが、これには対応していない。
- 拡張子 .mmm を持つ Type 1 フォントメトリクスデータには未対応。このメトリクスファイル は、Multiple Master Type 1 フォントに使われている。

9.3.2. 使用方法

本ライブラリで、Type1 フォントを使用する場合、他のフォント同様に、pl_SetFont 関数でフォント名、ウェイト、斜体の指定を行い、pl_ShowTextU 関数で文字出力を行う。この pl_SetFont 関数での各指定値と、実際のフォントファイルとの対応を以下に示す。

1. .AFM ファイルを使用する場合

フォント名	.AFM ファイル内の FamilyName に対応する。
ウェイト	.AFM ファイル内の Weight 値に対応する。AFM のウェイトは文字列であるが、ここに "Bold"、"Demi"、または Bold を含む文字列が記載されている場合、bold(700)とみなす。それ以外の値はすべて、normal(400)とみなす。
斜体	.AFM の ItalicAngle に対応する。この値が 0 で無い場合は斜体、0 の場合は非

	斜体とみなす。

2. .PFM ファイルを使用する場合

フォント名	.PFM ファイル内の WindowsName に対応する。
ウェイト	.PFM ファイル内の dfWeight 値に対応する。dfWeight は 400 または 700 を持つ。
斜体	.PFMdfItalic に対応する。この値が 0 で無い場合は斜体、0 の場合は非斜体とみなす。

9.3.3. フォントの名称指定

前項に記載したように、Type1 フォントのフォント名は、.AFM の FamilyName または.PFM の WindowsName から取得されるが、この２つは一致しないケースが多い。このため、Type1 フォントの名称指定には、いくつかの問題がある。

以下に実際の例を示す。

.PFB name (拡張子 略)	PFM 情報			AFM 情報			
	Windows Name	df Weight	df Italic	FullName	Family Name	Weight	Italic Angle
EU_____	Eurostile	400	0	Eurostile Medium	Eurostile	Medium	0
EUB_____	Eurostile Bold	400	0	Eurostile Bold		Bold	0
EUEX____	Eurostile ExtendedTwo	400	0	Eurostile Extended #2		Roman	0
EUBEX___	Eurostile ExtendedTwo	700	0	Eurostile Bold Extended #2		Bold	0

.PFM+.PFB の組み合わせで使用する場合、.PFM ファイルの名称は WindowsName で処理され、これらはすべて異なるため、指定は明確である。

一方、これらのフォントを、.AFM+.PFB の組み合わせで使用する場合、以下の状態となることから、指定は不明確なものとなる。

すべてのフォントの FamilyName が Eurostile である。また、ウェイトは Medium と Roman が font-weight="400" として解釈される。このため、フォント名= “Eurostile”、ウェイト=”400”が 2 種類、フォント名=”Eurostile”、ウェイト="700" が 2 種類存在することになる。

この問題を回避するための方法が 2 種類存在する。

1. WindowsName モード

   WindowsName モードでは、pl_SetFont で指定された名称と、PFM の WindowsName が合致すれば、このフォントが使用される。また、指定フォルダ内に.AFM が存在した場合も、その FamilyName は無視される。前述したとおり、.PFM の WindowsName モードでは名称の重複は発生しないため、上記の問題は回避できる。

なお、.PFM と同じベース名を持ち、拡張子.AFM のファイルが存在する場合、フォント名称以外の情報は.PFM に代えて、.AFM ファイル内のものを使用する。この処理により、.AFM ファイルをあわせて格納した場合、後述する.PFM ファイル使用時の制限次項は発生しない。

2. フォント設定ファイルによる別名定義

これはフォント設定ファイルで、FamilyName の重複が発生するフォントに対して、別名を定義する方法である。別名を定義した場合、元の FamilyName に優先するため、定義した別名を pl_SetFont で指定することにより、重複を回避することができる。

以下は、上記の例において、それぞれの AFM に WindowsName 相当の別名を定義する例である。

```
<font-config>
  <font-folder path="/home/resource/fonts">
    <!-- Set the family-name and weight to the PFM definition -->
    <font-alias   file="EU______.AFM">
      <alias family-name="Eurostile"/>
    </font-alias>
    <font-alias   file="EUB_____.AFM">
      <alias family-name="Eurostile Bold" weight="normal"/>
    </font-alias>
    <font-alias   file="EUEX____.AFM">
      <alias family-name="Eurostile ExtendedTwo"/>
    </font-alias>
    <font-alias   file="EUBEX___.AFM">
      <alias family-name="Eurostile ExtendedTwo" weight="bold"/>
    </font-alias>
  </font-folder>
</font-config>
```

9.3.4. 文字コードとグリフの対応

本ライブラリでは、pl_ShowTextU 関数で文字出力を行うが、ここで指定される文字コードの値と Type1 フォントのグリフの対応を以下に示す。

1. .AFM ファイルを使用する場合

Type1 フォントでは各グリフに、名称が定義されており、pl_ShowTextU 関数に入力された文字コードをこれと対応づける必要がある。Unicode とこのグリフの名称の対応付けは Adobe 社の Adobe Glyph List(AGL)に定義されている。本ライブラリでは、まず、pl_ShowTextU 関数に指定された Unicode 値から、AGL を参照してグリフ名を取得する。次に.AFM に記述されている、そのフォントのエンコーディングに関する定義を参照し、グリフ名から PDF に出力する文字コード(PDF 仕様書 文字セットとエンコーディングに記載)を決定して、このコードを出力する。

2. .PFM ファイルを使用する場合

.PFM ファイルは、PFM ヘッダの dfCharSet エントリに、エンコーディングデータを持つ。この 1 バイトのエントリには、文字セット(character set)と呼ばれる値が含まれる。Windows 環境では、WINGDI.H ファイルに、以下の文字セットが定義されている。

名前	値	コードページ
ANSI_CHARSET	0	1252
HEBREW_CHARSET	177	1255
ARABIC_CHARSET	178	1256
GREEK_CHARSET	161	1253
TURKISH_CHARSET	162	1254
VIETNAMESE_CHARSET	163	1258
THAI_CHARSET	222	874
EASTEUROPE_CHARSET	238	1250
RUSSIAN_CHARSET	204	1251
BALTIC_CHARSET	186	1257

Microsoft 社が提供する Unicode to code page mapping data を用いて、Unicode から文字コードへ変換し、PDF に出力する。コードページは 8 ビット文字幅のみを提供するため、このマッピングデータは最大で 256 個のエントリを持つことになる。

このため、下記のような制限事項が発生する。

コードページデータに定義されていないグリフは、フォントのアウトラインデータにグリフが定義されていても使用できない。

たとえば、Type1 フォントの標準エンコーディングである StandardEncoding の場合、LSlash、breve、dotaccent といったグリフが存在する。これは AGL ではそれぞれ、U+0141、U+02D8、U+02D9 という Unicode 値に対応が定義されている。しかし、上記の ANSI_CHARSET では、これらの Unicode が定義されていないため、脱落することになる。

また、コードページマッピングとフォントファイル内の実際のエンコーディングは、適合しない場合があり、.PFM+.PFB のペアで Type 1 フォントを使用することは、推奨できない。

9.3.5. 文字コードとグリフ名の対応付けの変更

前項目に記載した本ライブラリの Type1 フォントとグリフの対応付けを変更する方法を説明する。

前項に記載した対応付けは、一般に使用されている Latin 文字に対応するものであるが、フォントの中には AGL に適合しない特別なフォントも存在する。例えば、Adobe Type 1 製品である Carta (CR_____.AFM, CR_____.PFM, CR_____.PFB) には、189 の絵グリフと 標準外のグリフ名が定義されている。これらのグリフ名を AGL から調べると、適合するグリフ名は 14 個のみであり、それ以外は AGL に適合していない。このため、そのままでは、.AFM+.PFB の組み合わせの Carta は、ほとんどのグリフが使えないことになる。

この問題を回避する方法が 2 種類提供される。ひとつは、このフォント独自の グリフリストファイルを作成する方法、もう一つは、フォント設定ファイルに、<skip-glyphname-mapping>を指定する方法である。

1. グリフリストファイル

   このグリフリストファイルは単純なテキストファイルで、特定のフォントに対する Unicode とグリフ名のマッピングを記述する。フォーマットは、AGL ファイルと同様である。

   最初の項目は、4 桁の大文字 16 進数字で、Unicode の値を表す。次の項目は、.AFM ファイルに定義されているグリフ名である。３番目の項目は、Unicode の文字名であり、この項目は記載がなくても構わない。

   すべての項目はセミコロン ";" を使って分離されている必要がある。また、"#" で始まる行はコメント行とみなされる。

   以下にグリフリストファイルの一例を示す。このグリフリストファイルは、Unicode のプライベートユーザエリアを Carta フォントのグリフ名にマップするものである（ただし、空白と数字はそのまま）。

```
# Carta sample glyphlist file
# file name:carta-glyphname.txt
0020;space;
E000;circle;
E001;lookoutcontrol;
E002;triangle;
E003;diamond;
E004;hexagon;
E005;explode2;
E006;lookout;
E007;IRBM;
E008;ICBM;
E009;explode1;
E00A;ruin;
E00B;goldbar;
E00C;lighthouse;
E00D;mining;
E00E;gaging;
0030;zero;
0031;one;
0032;two;
0033;three;
0034;four;
E00F;boundary;
...
```

   以下は、このグリフリストファイル(carta-glyph-list.txt とする)をフォント設定ファイルに登録する例である。Carta フォントが c:¥FontsFonlder フォルダにあるものとする。

```
<font-config>
  <font-folder path="c:¥FontsFolder">
    <glyph-list file="carta-glyph-list.txt" afm="CR_____.AFM"/>
  </font-folder>
</font-config>
```

なお、標準 14 フォントの一つである ZapfDingbats も Carta 同様に AGL には Unicode のコード値との対応が定義されていない。本ライブラリに添付さえる ZapfDingbats-glyphname.txt は、フォントベンダが定義する各グリフと Unicode 値との定義から作成したグリフリストのサンプルである。

2. グリフマッピングのスキップ

これは、フォント設定ファイルに、<skip-glyphname-mapping> 要素を指定する方法である。

```
<font-config>
  <font-folder path="[Install directory]/fonts">
    <glyph-list file="zapfdingbats-glyphname.txt" afm="ZapfDingbats.afm"/>
  </font-folder>
  <font-folder path="/home/resource/fonts">
    <skip-glyphname-mapping afm="CR_____.AFM"/>
  </font-folder>
</font-config>
```

.AFM ファイルに対してこのオプションが指定されると、pl_ShowTextU 関数に指定された文字コード値が、フォントエンコーディングの範囲内にある場合、すべてそのまま PDF の文字にマップされる。例えば、指定されたコード値が 0x0021 の場合、この文字は Carta フォントのエンコーディングでは、10 進数の 33 が "circle" として定義されているため、直接 PDF ファイルに出力される。コード値 0x0101 は、Carta フォントのエンコーディングに定義されていないため、ミッシンググリフとしてエラーになる。
使用可能な文字コードは、.AFM ファイルで確認することができる。以下は、Carta フォントの .AFM ファイルの一部である。"C" の右側の数字と一致しているコード値を出力することにより、その文字が PDF に格納される。

```
EncodingScheme FontSpecific
StartCharMetrics 189
C 32 ; WX 280 ; N space ; B 0 0 0 0 ;
C 33 ; WX 560 ; N circle ; B 30 150 530 650 ;
C 34 ; WX 620 ; N lookoutcontrol ; B 15 60 605 741 ;
...
C 251 ; WX 852 ; N portofentry ; B 30 123 822 677 ;
C 252 ; WX 946 ; N whwycounty ; B 0 -58 946 857 ;
C 253 ; WX 1154 ; N whwytridown ; B 0 -100 1154 899 ;
C 254 ; WX 1072 ; N whwytriright ; B 0 -121 1073 919 ;
EndCharMetrics
```

9.3.6. 埋め込み

埋め込みを行わない場合は、メトリクスファイルだけが存在すれば PDF の作成が可能である。また、PDF の標準 Type1 フォント 14 種(Courier、Courier-Bold、Courier-Oblique、Courier BoldOblique、Helvetica、Helvetica-Bold、Helvetica-Oblique、Helvetica-BoldOblique、Times-Roman、Times-Bold、Times-Italic、Times-BoldItalic、Symbol、ZapfDingbats)は埋め込みを行わない場合も、各 Reader で表示が保証されるものである。本ライブラリではこれらのフォントの埋め込みは指定の有無にかかわらず行っていない。
フォントの埋め込みを行う場合、アウトラインデータが必要となる（具体的には .PFB ファイルである）。.AFM あるいは.PFM だけが存在するフォント環境で埋め込みを指定した場合、エラーとなる。

現在、本ライブラリでは Type1 フォントの埋め込みについてはは、すべての文字の埋め込み（フルセット埋め込み）としている。指定された文字だけの埋め込みはサポートしていない。

9.4. TrueType フォント、TrueType アウトラインを持つ OpenType フォント

9.4.1. 概要

OpenType フォントには TrueType アウトラインを持つフォントと PostScript アウトラインを持つフォントの 2 種類が存在する。TrueType フォント、および TrueType アウトラインを持つ OpenType フォントは通常、拡張子に.TTF または .TTC を持つ（OpenType フォントは拡張子.OTF も認められている）。PostScript アウトラインを持つ OpenType フォントは通常、拡張子に.OTF を持つ。いずれも Type1 フォントとは異なり 1 つのファイルから構成される。
ここでは、TrueType フォントと TrueType アウトラインを持つ OpenType フォント(以下、まとめて TrueType フォントとする)について記述する。

本ライブラリで使用できる TrueType フォントは、Unicode からグリフインデクスへの cmap を持つフォント、または Symbolic フォントのいずれかである。以下、簡単に説明する。
TrueType では文字コードから、グリフインデクスへの対応にフォントに内蔵される cmap を使用する。この cmap には、Unicode からグリフインデクスへ、あるいは Shift-JIS からグリフインデクスへ、というようないくつかの形式が存在し、通常のフォントでは、これらを複数内蔵し、使用する側が必要なものを参照できるようになっている。本ライブラリでは、Unicode からグリフインデクスへの cmap を参照している。
現在、一般的に使用されているフォントには Unicode からグリフインデクスへの対応表が定義されている。
また、Symbolic フォントは、Symbol、Wingdings といったフォントであり、上記とは異なり、1 つだけ cmap を持つフォントである。

9.4.2. 使用方法

本ライブラリで、TrueType フォントを使用する場合、他のフォント同様に、pl_SetFont 関数でフォント名、ウェイト、斜体の指定を行い、pl_ShowTextU 関数で文字出力を行う。この pl_SetFont 関数での各指定値と、実際のフォントファイルとの対応を以下に示す。

フォント名	次の値を持つ name table データに対応する。 ● Platform ID = 3 (Microsoft) ● Platform-specific encoding ID = 1 (Unicode) ● Language ID = 0x409 (English - United States) ● Name ID = 1 (Font Family Name)
ウェイト	OS/2 table の usWidthClass エントリの値に対応する。 このエントリには、100～900 までの 100 単位の太さの値が定義される。
斜体	OS/2 table の fsSelection エントリの最下位ビットに対応する。このビットがオンならば、font-style="italic" とみなされる。

別の言語 ID と共に複数のフォントファミリー名を持つフォントがある。フォント名にはこれらの名前を使うこともできる。例えば、simsun.ttf は、"SimSun" と "宋体" という 2 つのフォントファミリー名を持つが、

どちらの名前で指定することも可能である。

9.4.3. 埋め込み

本ライブラリではTrueType フォントのアウトラインをPDFファイルへ埋め込むことができる。TrueType フォントは、OS/2 table の fsType エントリに、埋め込みに関するライセンス情報を持っており、この設定によってフォントベンダが埋め込みの禁止を指定することができるようになっている。本ライブラリではこのライセンス情報を参照し、埋め込み指定されたフォントが埋め込み禁止フォントであった場合、エラーを戻す。
TrueType フォントの埋め込みは、使用されているグリフだけを埋め込むサブセット埋め込みとなる。

9.4.4. その他

TrueType、OpenType では、Advanced Typographic Feature と呼ばれる機能が定義されている。本ライブラリではこの機能のうち、GSUB テーブル（Glyph Substitution Table）の vert フィーチャーのみをサポートしている。
vert フィーチャーは縦書き時に別のグリフを通常（横書き用）グリフとは異なるグリフを割り当てる場合にそのグリフ番号を指定するものであり、主に CJK フォントで使用される。

9.5. PostScript アウトラインを持つ OpenType フォント

9.5.1. 概要

PostScript アウトラインを持つ OpenType フォントは通常、拡張子に.OTF を持つ。前項の TrueType 同様、１つのファイルから構成される。
PostScript アウトラインを持つ OpenType フォント は、OpenType (PostScript) CID フォントと、OpenType (PostScript) non-CID フォントの２つのカテゴリに分類される。

Non-CID フォント	主に Latin 文字のグリフを含み、グリフは、グリフ名を使ってインデクスされる。PDF には Type1 フォントとして出力される。
CID フォント	主に CJK ideograph グリフを含み、グリフは、CID を用いてインデクスされる。PDF には Type0 (CIDFontType0)として出力される。

9.5.2. 使用方法

pl_SetFont 関数で指定されるフォント名、ウェイト、斜体指定と実際のフォントファイルとの対応は TrueType と同様である。
なお、ウェイトに 100 刻みでない値を持つフォントがあった場合、100 刻みの近い値を使用することで対応している。
その他、埋め込みなどに関しては、TrurType と同様である。

10. イメージ出力について

10.1. サポートされるラスターイメージフォーマット

本ライブラリから PDF 出力が可能なラスターイメージ形式は、BMP,JPEG,PNG,TIFF,GIF である。それぞれに関する注意事項を以下にまとめる。

形式	一般的な拡張子	サポートされる形式、および制限事項に関する説明
BMP(Windows Bitmap)	.bmp	1,4,8,16,24,32 ビットカラーのビットマップ形式
JPEG	.jpg、.jpeg	JFIF : Jpeg File Interchange Format Adobe Photoshop の CMYK JPEG ファイル ベースライン圧縮のみ対応
PNG(Portable Network Graphics	.png	サポートされる各形式 ● ColorType が Grayscale,TrueColor の tRNS チャンク付きイメージは tRNS チャンクの内容を透過色として出力する。 ● ColorType が Index の tRNS チャンク付きイメージは tRNS チャンクの内容が連続した 1 つの範囲のみ完全透過であり、その他がすべて完全に不透過の設定となっていれば、透過色として出力する。そうでない場合はαチャネル付きイメージとして出力する。 ● ColorType がα付きの Grayscale,TrueColor の場合はあるαチャネル付きイメージとして出力する。 以下の制限事項が存在する。 ● PDF1.3 では、αチャネルは出力できない。上記でαチャネル付きイメージとして出力する、と記載した形式は、pl_SetTransImgProcMode 関数で PL_Transparency__NotProc が指定されている場合、pl_CheckImage にて未サポートが戻される。PL_Transparency_NeetProc が指定されている場合、PL_CheckImage ではサポートイメージとして扱い、αチャネルが除去された(不透過)形式で出力される。 ● Grayscale および TrueColor の成分が 16 ビットの場合、8 ビットに切り詰めて出力する。
TIFF(Tagged Image File Format)	.tif,.tiff	TIFF Rev 6.0 仕様記載の形式をサポートする。以下に説明する。 ● カラータイプ バイレベルイメージ、グレースケールイメージ、パレットカラーイメージ、RGB フルカラーイメージ、CMYK イメージをサポートする。 （CIE L*a*b* は未サポート） ● 圧縮形式 非圧縮、Modified Huffman、

		CCITT Group 3 1-D,CCITT Group 3 2-D,CCITT Group 4、LZW 圧縮、PackBits、JPEG および Photoshop ZLIB 圧縮をサポートする ● カラー成分が 16 ビットの場合、8 ビットに切り詰めて出力する。 ● αチャネルの取り扱いについては、PNG と同様である。
GIF(Graphics Interchange Format)	.gif	透過色は PDF の透過色として出力する 以下の制限事項が存在する ● アニメーション GIF は最初のイメージのみ出力、 ● プレーンテキスト拡張は未サポート ● 背景色はサポートしない
JPEG2000	.jp2 など	JP2 形式 PDF1.5 以上ではパススルーにて出力する。PDF1.4 以下では、一旦解凍後、出力を行う。 JasPer に依存する。

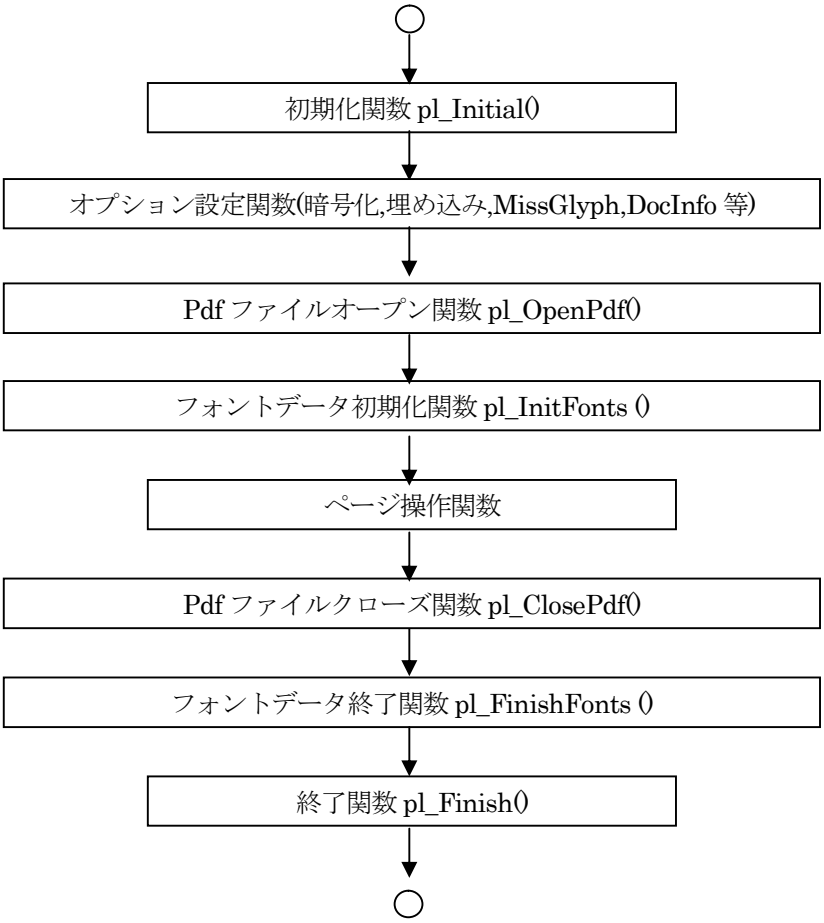
イメージ形式については内部で自動判定を行い、適当な形式で出力を行う。
対応可能な形式か否かについては pl_CheckImage 関数により事前に判定を行うことができる。未サポート形式であった場合、呼び出し側で BMP などの上記のサポート可能な形式に変換するといった手段がとれれば、その形式に変換して出力することができる。

10.2. イメージパススルー

一般的にイメージデータはなんらかの圧縮が行われている場合が多い。この圧縮形式と、元のイメージデータの形式（カラースペース、色数など）が共に PDF でサポートされているものである場合、本ライブラリでは受け取ったイメージデータをそのまま PDF に格納する方式を取っている（これをイメージパススルーと呼ぶ）。これにより、イメージデータの高速な出力が可能である。
ただし、判別などに必要となる部分以外のデータの確認処理は行っていないため、破損したイメージデータが出力された場合、ビューアが参照できない、という状態が発生しうる。
いずれかが PDF でサポートできない形式の場合、本ライブラリでは内部で圧縮を解き、PDF に互換のビットマップイメージ形式に変換を行った後、ZLIB または JPEG 圧縮を行って、PDF に格納する。
圧縮設定関数 pl_PutCompressOpt の ColorImage、および JpegQuality で設定するオプションは主に BMP 形式のイメージデータが入力された場合のためのものであるが、上記の解凍をおこなったイメージの PDF への格納時にも参照される。
また、本ライブラリが JPEG 圧縮をサポートしていないイメージ形式（カラーパレットを使用したイメージなど）は指定にかかわらず、ZLIB 圧縮を行う。

11. サンプル

11.1. 呼び出し方法



12. 商標と著作権情報

12.1. 商標

Adobe、Acrobat、および Reader は、アドビ システムズ社の米国ならびに他の国における登録商標または商標です。

Microsoft、Windows、OpenType は米国 Microsoft Corporation の米国およびその他の国における登録商標または商標です。

TrueType は米国その他の国で登録された米国アップルコンピュータ社の商標です。

その他記載されている全ての会社名および製品名は、個々の所有者の登録商標または商標です。

12.2. 著作権情報

本ソフトウェアは、以下のライブラリを使用する。

This product includes software developed by Independent JPEG Group(http://www.ijg.org/).

Copyright (c) 1991-1998 Thomas G. Lane.

This product includes software developed by Sam Leffler and Silicon Graphics, Inc
(http://www.libtiff.org/).

Copyright (c) 1988-1997 Sam Leffler

Copyright (c) 1991-1997 Silicon Graphics, Inc.

This product includes software developed by Jean-loup Gailly and Mark Adler
(http://www.gzip.org/zlib/).

Copyright (C) 1995-2002 Jean-loup Gailly and Mark Adler

This product includes software developed by Glenn Randers-Pehrson and many other contributors
(http://www.libpng.org/pub/png/).

Copyright (c) 2000-2002 Glenn Randers-Pehrson

Copyright (c) 1998, 1999 Glenn Randers-Pehrson

Copyright (c) 1996, 1997 Andreas Dilger

Copyright (c) 1995, 1996 Guy Eric Schalnat, Group 42, Inc.

This product includes altered version by Antenna House,Inc.

This product includes softwares developed by:

International Business Machines Corporation

International Components for Unicode (ICU) libraries

http://www.icu-project.org/

Copyright (c) 1995-2010 International Business Machines Corporation and
others All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish,distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT,NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder.

This product includes softwares developed by:
The FreeType Project is copyright (C) 1996-2000 by David Turner, Robert Wilhelm, and Werner Lemberg.

This product includes softwares developed by:
Little cms http://www.littlecms.com/
LittleCMS 1.13 Copyright (C) 1998-2004 by Marti Maria

This product includes softwares developed by:
Image Power, Inc. and many other contributers.
http://www.ece.uvic.ca/~mdadams/jasper/
JasPer 1.701
Copyright (c) 1999-2000 Image Power, Inc.
Copyright (c) 1999-2000 The University of British Columbia

Copyright (c) 2001-2003 Michael David Adams

This product includes softwares developed by:
The Code Project http://www.codeproject.com/cpp/yard-tokenizer.asp
A Regular Expression Tokenizer using the YARD Parser
distributed under the Boost Software License, Version 1.0.
(http://www.boost.org/LICENSE_1_0.txt)
Copyright 2004 by Christopher Diggins
Everything else Copyright (c) CodeProject,1999-2009

以上