特集「電子出版・電子新聞」

5. 電子出版の一手法・PDF

川又行雄（アドビシステムズ（株））

InternetやCD-ROM上でAdobe PDFを目にする機会も多くなってきている．PDF（Portable Document Format）は，Adobe Acrobat製品が処理するデータ形式である．本稿では，このAdobe PDF（以下PDFと記す）と，これを扱うAdobe Acrobat（以下Acrobatと記す）について述べる．

PDFの特長

PDFの第1の特長は，その名前が表す通り，ポータブル（携帯性）にある．一般的にポータブルラジオなどの言葉で使われる「ポータブル」には，「持ち運びが簡単」というイメージがある．そのためには，「小型軽量」，「使用する場所を選ばない」などの性質を持っている必要がある．PDFは，書類を表すデータ形式としてこれらの性質を持っている．

□ 小型軽量

データそのものが圧縮の機能を持っているのでデータの大きさを小さくすることができる．AcrobatでPDFを表示する際にページごとに伸長しながら表示を行っている．「圧縮」に関しては後述する．さらに，CPUの高性能化やキャッシュの技術によって，それほど「重さ」を感じさせない操作性を提供している．

□ 使用する場所を選ばない

ここで言うところの「場所」とは「PDFを処理するコンピュータ」を意味している．Window，Mac OS，UNIX上でPDFを扱うことができる．PDFを閲覧するAcrobat Readerは無料で配布されている現在はUNIX上の日本語表示機能の提供が遅れているが，近い将来これも実現することが検討されている．

その他，PDFの特長として以下の項目を挙げることができる．

□ PDFを作成するソフトを選ばない

Macintosh，Windows上の印刷機能を持ったソフトであれば，その印刷イメージをPDFにすることができる．またUNIX上のソフトに多く見られるようなPostScriptを生成するソフトに対してもMacintosh，Windows同様に印刷イメージをPDFにすることができる．これに関する詳細は「PDFの作成」で述べる．

□ 描画の再現性が高い

PDFは，Adobe PostScriptで培われたフォントとグラフィックスの技術を継承している．これによりプラットフォームに依存しない画面表示と，PostScript印刷を実現している．PICT印刷（Macintosh），GDI印刷（Windows）にも対応しているが，PDFとPostScriptの親和性を考えるとPostScript印刷の方がより高いアドバンテージを受けることができる．また，デバイスに依存しないカラー（DIC: Device Independent Color）にも対応している．

□ Webシステムとの融合性

Internet上で広く普及しているWebシステムとの融合性が高いことも注目すべき点である．MS Internet Explorer，Netscape CommunicatorなどのWebブラウザと共にAcrobatを使用することにより各ブラウザのウィンドウ内にPDFを表示する機能を提供している．これによりWebブラウザの利用者に対し高い操作性を提供している．さらにHTTPのバイトサービング機能にも対応しているので，PDF内の全ページのダウンロード終了を待つことなしに，見たいページごとのダウンロードだけを表示することができる．これによりネットワークの負荷を軽減することができる．また，表示されたページは，マウス操作で拡大縮小が自由自在である．

□ ハイパーテキスト機能

ドキュメントの閲覧に対して，さまざまなリンク機能を提供している．

- 同一PDF内の任意のページ
- 他PDF内の任意のページ
- 他のアプリケーション（MS Word，MS Excelなど）で作成された書類
- QuickTimeを用いた動画，サウンド
- Webシステムと共に使用したURL

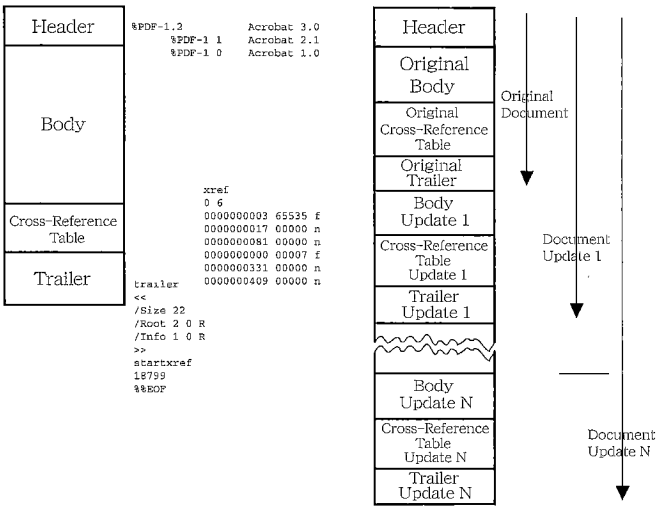

図-1　PDF File Structure

□ セキュリティ

パスワードの設定によって次のような各機能別の使用制限を書類ごとに付加することができる．

- 書類の開示
- 書類の変更
- 印刷
- テキストやグラフィックスの選択
- ノートの追加や変更

□ 拡張性

Acrobatの拡張性と関連することであるが，PDFはデータとしての拡張性もある．ユーザ定義のデータ部分を定義してそれを独自のPlug-insモジュールで処理することも可能である．システム開発者向けに提供されているAcrobat SDK（Software Development Kit）を用いればこれを実現することも可能である．

■ PDFの構造 ■

PDFはPostScriptのグラフィックスの概念を継承しているが，その性質はかなり異なっている．

PostScriptは完全なプログラミング言語であり，実行制御，スタック操作，算術演算，論理演算，および描画などのオペレータ（命令）を持っている．またPostScript自体は構造を持っていないためそれを補うためにDSC（Document Structuring Convention）というPostScriptのコメント行を利用した構造定義の枠組みが用意されている．このDSCをさらに発展させたのがEPSF（Encapsulated PostScript Format）である．これは，DTP（DeskTopPublishing）を利用している人にはお馴染みのデータである．PostScriptはプログラミング言語であるために，プログラマ以外の人には使い難い．そこで登場したのがAdobe Illustratorである．Illustratorのデータは PostScriptに制限を加えて実現している．Illustratorによりプログラミングではなくマウスやキーボードの操作によりPostScriptのデータが作成できるようになった．EPSFをPostScriptのプログラミングにより作成している人はほとんどいないであろう．大部分の人がIllustratorまたはAdobe Photoshopを利用してEPSFを作成しているであろう．Illustratorで作成されたEPSFは，描画だけに重点をおいているのでPostScriptの実行制御，スタック操作，算術演算，論理演算をほとんど用いていない．Photoshopで作成されたEPSFも同様にimageオペレータを用いたPostScriptのビットマップデータである．PDFの描画命令は，これらIllustratorとPhotoshopのデータをヒントにデザインされている．

PDFの構造は，大きく「ファイル構造」と「書類構造」に分けることができる．そしてこれらを構成する要素である「オブジェクト」とPostScriptを継承した「ページ記述」などからPDFは成り立っている．

□ ファイル構造

図-1の左側の部分を参照しながら述べる．

PDFは以下のファイル構造を持つ．

Header
Body
Cross-reference Table
Trailer

Header部は単純にPDFのバージョン情報を持った以下の行だけからなる．

%PDF-1.2

バージョン番号はAcrobat 3.0では1.2であるが，Acrobat 2.1の時は1.1，Acrobat 1.0の時は1.0であった．

Trailer部は，階層構造を持ったBody部の先頭（Root）と書類情報（Info）のオブジェクト番号，さ

らにクロスリファレンステーブルの開始場所（startxref）を持つ．具体例を次に示す．具体例では，見慣れない表記が用いられているが，必要に応じて理解に必要な表記の意味を順次説明する．

キーワードtrailerから始まり，ファイルの終了を示す％％EOFで終わる．<<...>>はPostScript Level2でお馴染みのkey-value pairをデータ構造に持つ辞書を表す．キー/Sizeの値は22．構造の始まりである/Rootは，間接オブジェクト（Indirect objects）の2（2 0 R）を参照，文書情報（/Info）は間接オブジェクトの1（1 0 R）を参照することを表している．間接オブジェクト（Indirect objects）とは，オブジェクトの実体が後で定義されることを示している．PDFでは，通常のプログラミング言語における変数の概念がないので，間接オブジェクトにより参照関係を実現している．n 0 Rは参照を示し，これの実体はn 0 obj ... endobjで定義される（ここでnは整数）．

Cross-Reference Table部は，PDFで使用される間接オブジェクトをアクセスするための，ファイル内のバイト位置情報を持つ．Acrobatはあらかじめこの Cross-Reference Tableを読み込んでおき，間接オブジェクトの参照が必要になると，このテーブルを経由してオブジェクトの実体を得る．間接オブジェクトはシンボルではなく番号で参照される．図におけるxrefの部分に注目していただきたい．

この例では，0から始まる6個，つまりオブジェクト番号0から5までのリファレンス情報を表している．右端にfがついたオブジェクト0と3はフリーオブジェクトで使用されていない．ただし3を参照した時は7を見よという意味である．PDFの変更が加えられるとこのように参照を変更する．左側の10桁の10進数はファイル内の先頭からのバイトオフセット，中程の5桁の10進数は代わりのオブジェクト番号を表す．右端にnがついたオブジェクト1，2，4，5はそれぞれこのファイルの17，81，331，409バイト目にその実体があることを表している．

Body部は，PDFの「書類構造」そのものであるので後述する．

さて，Acrobatなどを使用して既存のPDFに対して変更が加えられた場合，通常は変更のあったオブジェクトに対してのみ追加的な変更が行われる．これをIncremental updateという．新たに追加されたオブジェクトは単純に追加，削除されたオブジェクトは参照情報でフリーに，変更の場合は旧オブジェクトをフリーにして新オブジェクトが追加される．図-1の右側で表すように，更新を1回行うとBodyのオブジェクト，Cross-Reference Tableのオブジェクト，Trailerのオブジェクトが追加される．このように何回か変更作業を行うとPDFは図のようになって，ファイルサイズが増え続けていく．Acrobatはその時点で表示に必要な情報だけしかメモリ内に読み込んでいない．それ故に大きなサイズのPDFでもそれほどメモリを消費せずに表示することができる．PDFファイルが変更された場合，変更に影響があるオブジェクトだけが保存の時に追加書き込みされ，保存の際の時間の節約を行っている．

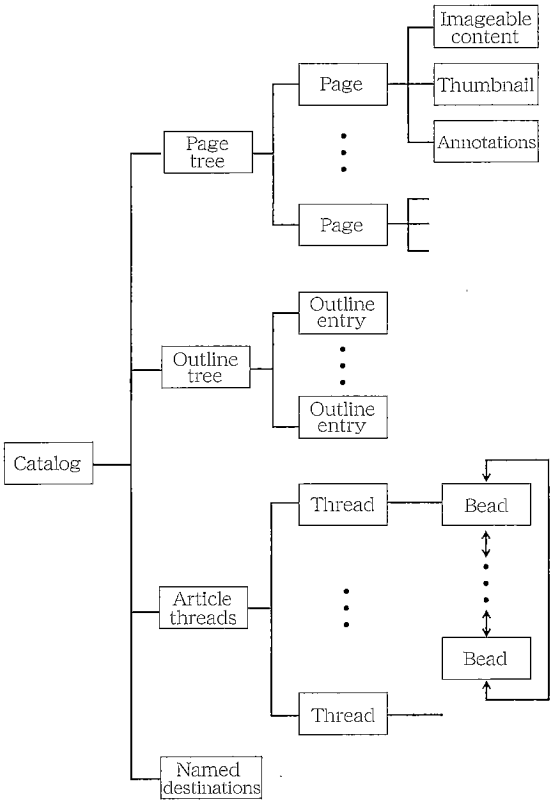

図-2　PDF Document Structure

最終的なPDFの変更が終了した時，またはWebで公開するPDFを作成した場合は，必ず「最適化」の作業を行う必要がある．「最適化」によりHeader，Body，Cross-Reference Table，Trailerの再構築を行い，不用オブジェクトの物理的な削除，さらにテキスト，グラフィックス，イメージに対する最適化も行っている．「最適化」により線形化（Linearize）の処理も行われている．

PDFのファイル構造を見た時，「Cross-Reference TableとTrailerが物理的にファイルの後にあると，必ずその部分を読み込まないとページの表示ができないから，Webのバイトサービングはできないのではないか？」と思われる方もいるであろう．その答えは線形化で解決される．線形化されたPDF（Linearized PDF）は，Cross-Reference TableとTrailerを含むPDF全体を管理する情報をファイルの前の方に位置付ける処置を行う．

□ 書類構造

PDFの書類構造を図-2に示す．

図における各ボックスは1つのオブジェクトである．例としてカタログ（Catalog）のオブジェクトを次に示す．

```
1 0 obj
<<
/Type /Catalog
/Pages 2 0 R
/Outlines 3 0 R
/PageMode /UseOutlines
>>
endobj
```

objの行から始まってendobjまでが1つの間接オブジェクトである．オブジェクト番号は先頭の1，次の0は有効なオブジェクトであることを表すGeneration番号．タイプ（/Type）はカタログ（/Catalog）で，ページを管理するオブジェクト（/Pages）は間接オブジェクトの2を参照，Outlines（Acrobat 3.0Jでは「しおり」）は間接オブジェクト3を参照，PDFを開いた時（/PageMode）は「しおり」と共に表示する（/UseOutline）ということを意味している．

図におけるArticle threadsはAcrobat 3.0Jの「アーティクル」を表す．これはテキストフローの情報を持つ．Named destinationsは，リンク先のページ位置に名前をつけておいて，名前によるリンク参照や<http://www.adobe.com/xyz.pdf#nameddest=Chap2.Section3>のようなWeb上でのページ指定の環境を提供する．HTMLにおけるNameと同様である．Pages treeはPDF内のページを管理していて，その下にあるPageが各1ページを表している．Page内の各オブジェクトは次に示す通りである．

Imageable content

この部分がPostScriptの性質を継承した「ページ記述」である．テキスト，グラフィックス，イメージの描画命令が記述されている．通常は圧縮，暗号化されたstreamオブジェクトになっているので，Acrobat以外でPDFを強制的に見ても意味が分からない文字列になっている．

Thumbnail

ページをラスタライズした結果の縮小イメージがサムネール（親指の爪）として格納される．

Annotations

リンク，ノートなどの付加情報を格納する．

このようにPDFはさまざまな情報を付加できる可能性を持っている．しかしながらPDFを意識しないでPDFを作成した場合，つまり任意のアプリケーションから「印刷機能」を用いて作成されたPDFはCatalog→Pages tree→Pages→Page→Imageable contentで示すデータとPageの繰り返しになる．これと対照的に，PDFを意識して作成されたPDF，つまりAdobe PageMakerやAdobe FrameMakerなどの「PDF作成」機能で作成されたPDFでは，印刷には含まれない「しおり」，「アーティクル」，「リンク」なども自動生成することが可能になる．

AcroForm機能を用いてFormのデータが付加された場合は，カタログ（Catalog）にそのエントリーである/AcroFormが追加される．次に例を示す．/AcroForm 9 0 Rを起点として追ってみよう．

```
1 0 obj
<<
/Type /Catalog
/Pages 2 0 R
/Outlines 3 0 R
/AcroForm 9 0 R　%間接オブジェクト9 0 objを参照
/PageMode /UseOutlines
>>
endobj
...
9 0 obj
<<
/Fields［77 0 R 76 0 R 78 0 R 84 0 R 85 0 R］
  %フィールドは5個.
/DR<< /Font<< /ZaDb 15 0 R /Helv 1 0 R
  /HeBo 23 0 R>> >>
/DA（/Helv 0 Tf 0 g）
>>
endobj
...
77 0 obj　%最初のフィールド
<<
%このオブジェクトのタイトルはsubmit
/T（submit）
/Kids［19 0 R］　%19 0 objを参照
/FT /Btn　%ボタンであることを表す
/Ff 65540
/DA（/HeBo 18 Tf 0 g）
>>
endobj
...
19 0 obj
<<
/Type /Annot
/Subtype /Widget
/Rect［439 173 473 346］
/F 4
/H /P
/AP<< /N 20 0 R /D 21 0 R>>
/Parent 77 0 R　%親オブジェクトは77 0を参照
/BS<< /W 1 /S /B >>
/A 83 0 R
>>
endobj
```

オブジェクト76，78，84，85は省略．

詳細な説明は省略するが，submitという名前のボタンが定義されている雰囲気は味わっていただけるで

あろう．これがテキストフィールドの場合は次のようになる．

```
77 0 obj
<<
/T (pdf_4)
/Kids [19 0 R]
/FT /Tx
/DR<< /Font<< /ZaDb 78 0 R /Helv 10 0 R
  /KaGo 16 0 R >> >>
/DA (/KaGo 0 Tf 0 g)
/AA 76 0 R
/TU (kawamata)
/V (1/3/98)
>>
endobj
```

これも雰囲気を味わっていただけるであろう．「角ゴシック」で表示（/DA），フィールドのタイトル（/T）はpdf_4，初期値（/V）は1/3/98である．これらはPDFのデータの一部である．AcroFormではsubmitでWebサーバへデータを送信したり，あるいはフォームのデータ部分だけを取り出すことができる．このデータ形式をFDF（Form Data Format）と呼ぶ．FDFの構文規則はPDFを継承している．前述のAcroFormに対するFDFは次のようになる．

```
%FDF-1.2
1 0 obj
<<
/FDF<< /Fields [
...
<< /V (1/3/98) /T (pdf_4) >>
...
]
/F (Form_Test.pdf)
...
>> >>
endobj
trailer
<<  /Root 1 0 R>>
%%EOF
```

PDFではAcroFormの表示に必要な，フォームの種別，フォント，座標，名前，初期値を持っているのに対して，FDFでは名前に対する値の情報を持つことになる．タイトル（/T）pdf_4の値（/V）は1/3/98を表している．このFDFを取り出したり作成するために，WebサーバのCGI（Common Gateway Interface）コマンドを作成するためのライブラリとしてAcrobat FDF Toolkitがある．

フォントについて

出版物を表現する場合，避けて通れない重要な課題にフォントがある．文書を作成した際に用いたフォントと同じフォントで文書を閲覧することが重要なポイントになる．

基本的に，PDFを表示するシステムで使用可能なアウトラインフォントを用いて文書の再現性を実現している．文書作成時に用いたフォントがないシステムでPDFを閲覧する場合，2つの回避方法がある．フォントの埋め込み（Font Embedding）と代替フォント（Font substitute）である．

埋め込めるフォントは，アウトラインフォントであるTrueTypeとPostScript Type1フォントである．またフォント全体を埋め込むのではなく，その文書内で使用されている文字のみを対象とした埋め込みサブセット機能も有している．フォントの埋め込み機能は，フォントのコピーとみなされるので，フォントの著作権者が許可を与えている範囲での使用に注意を払う必要がある．さらにPDFのファイルサイズにおけるフォントデータ部分の占有率，フォントの再利用性を考慮しつつフォント埋め込み機能は慎重に扱う必要がある．なお，Acrobat 3.0Jでは漢字フォントの埋め込み機能は提供されていないが将来は提供されることが検討されている．

文書作成時に使用されたフォントが，埋め込まれていない場合，かつAcrobatを使用しているシステム上にない場合，Acrobatは次のような代替フォントの処理を行う．ローマンフォントの場合，Acrobatと共にインストールされるマルチプルマスターフォントを用いて表示される．当該フォントの特徴に応じてサンセリフまたはセリフが使用される．漢字フォントの場合，漢字のマルチプルマスターフォントがないので，PANOSE属性による代替処理が行われる．そのシステムで使用可能なアウトラインフォントの中でPANOSE属性が最も近いフォントで代替される．

代替フォントの処理は，PDF作成時のフォントがない場合に行われる一種の救済処置である．「出版物」という物を扱う際には代替フォントが使用されないような考慮を施す必要がある．

PDF閲覧者が特定できる場合には，PDFの配布とは別に使用するフォントを，フォントの使用許諾に則った方法でPDF閲覧者のシステム上にあらかじめインストールしておくとよいであろう．

圧縮について

PDFのファイルサイズが小さいことは大きな特長の1つである．PDFでは各ページの「ページ記述」部分が圧縮されていて，各ページの描画の際に伸長処理

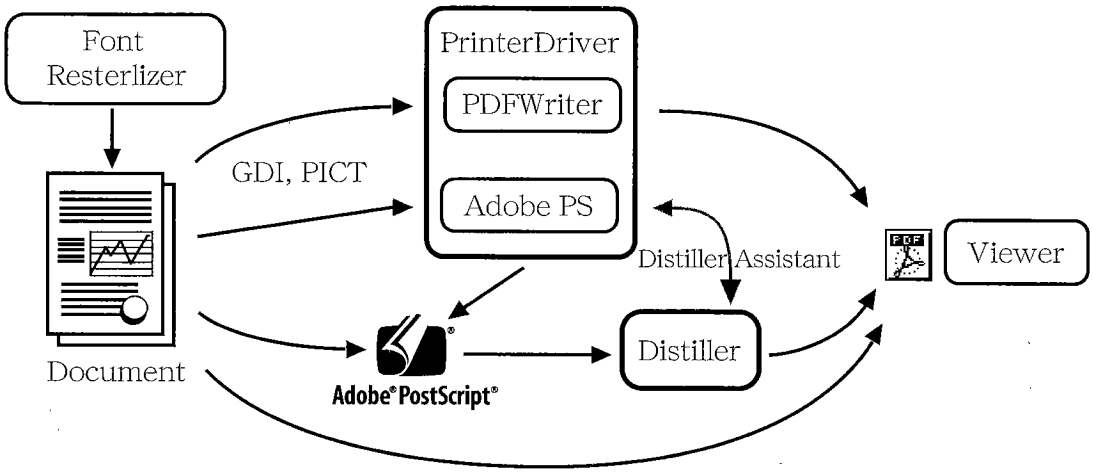

図-3　PDFの作成

が行われている．このことは，圧縮ツールなどにより全体が圧縮されている場合に比べて，表示の際に一度全体を伸張する必要がないので，時間的，容量的な節約を行え，PDFの使い勝手が良くなっている．さらにデータの特性に応じた圧縮方式を用いることにより，最もデータに合った圧縮方式を指定することができる．

PDFでは，テキスト，グラフィックス，イメージの各データがページ記述の部分に現れる．この内テキストとグラフィックスの描画は，PostScriptに似た描画命令の並びなので，記述は7bit ascii文字列である．この部分に関してはZIP圧縮が用いられている．Acrobat 2.1まではLZWが用いられていたがAcrobat 3.0ではZIPである．埋め込まれたフォントの部分もテキスト，グラフィックスと同じ圧縮方式が適用されている．

イメージの部分は，ページの記述の中からは間接オブジェクトとして参照され，イメージ本体はページ記述の本体とは独立して存在している．

このイメージには，カラー画像，グレースケール画像，白黒画像の3種類が存在し，それぞれに対して個々に圧縮方式を指定することができる．カラー画像，グレースケール画像に対しては，5段階のJPEG圧縮，ZIP（4bit），ZIP（8bit）を，白黒画像に対してはファクシミリでお馴染みのG3，G4およびZIP，RunLengthの圧縮方式を指定することができる．Distillerではこれらを指定することができるが，簡便なPDFWriterの場合は指定できる項目が少なくなる．

以降はAcrobat製品に依存した説明である．「圧縮」とは直接関係がないが，AcrobatではPDF作成時にイメージデータのサブサンプル，ダウンサンプル機能を持っている．いずれもカラー画像，グレースケール画像に対して，Macintoshでは72dpi，Windowsでは90dpi，白黒画像では300dpiになるようにピクセルを間引く処理が施される．これはデータサイズの減少に大いに寄与するが，印刷をすると思わぬ品質の劣化を招く．Acrobatの圧縮に関する設定にはデフォルト設定に任せないで用途に合った手動圧縮を指定するとよい．品質の劣化を防ぐことが再優先の場合やフルカラー（24bit）の場合，またスクリーンキャプチャやイラストのような色境界がはっきりした画像にはJPEGは避けるべきであろう．参考までにダウンサンプルとは，間引かれる領域の平均値のピクセル値を用いる方式であり処理に若干時間がかかり結果の画像は少しぼやけた感じがする．結果的にアンチエイリアシングを施した場合と同等の効果がある．サブサンプルとは，間引かれる領域の中心のピクセルを取り出す方式であり結果の画像はシャープな感じになる．

PDFの作成

PDFの作成にはいくつかの方法がある．**図-3**を参照しながらこれらについて述べる．

PDFWriterを使用する方式

図におけるDocument→PDFWriter→Viewerの流れである．MacintoshまたはWindowsで一般的に行われる方式である．プリンタを選択するのと同じ方法でAcrobatのPDFWriterを選び，通常のアプリケーションからプリント命令を実行する．この時，Macintoshの場合はPICT，Windowsの場合はGDIという各システムが標準で扱う描画データがプリンタドライバであるPDFWriterへ送られる．PDFWriterはプリンタのふりをしながらプリントの代わりにPDFを生成する．この方式が最も簡便なPDF作成方法である．

Adobe PostScript Printer Driverを使用する方式

図におけるDocument→Adobe PS→Adobe PostScript→Distiller→Viewerの流れである．

PDFWriterを使用する方式と操作は同様である．異なる点はプリンタ選択方法だけである．具体的な方法はMacintoshとWindowsでは若干異なるので詳細は製品のマニュアルを参照されたい．PDFWriterの場合と同様に，Macintoshの場合はPICT，Windowsの場合はGDIがプリンタドライバであるAdobe PSへ送られる．Adobe PSはAdobe PostScriptを生成し，それがAcrobat Distillerに渡され最終的にPDFが生成される．この時，Adobe PSとDistillerはDistiller Assistantによって制御されるので，使用者がAdobe PostScriptを見ることはない．PDFWriterの方式に比べ，事前にDistillerの設定を行うこととかPDF作成に時間がかかるなどのデメリットはあるが，より高品質のPDFを得ることができる．特にEPSFを貼り込んだ書類をPDF化する場合にはこの方式を用いると最良の結果が得られる．

□ PostScriptからPDFを作成する方式

図におけるDocument→Adobe PostScript→Distiller→Viewerの流れである．PostScriptを直接生成するアプリケーションに適用できる方式である．ネットワーク上のサーバ環境での構築も比較的容易に行える．さらにpdfmarkなどのDistiller用のPSオペレータをPS内に埋め込むことにより，印字に直接関係のないリンクやしおりなどのPDF閲覧に便利な機能を自動生成することも可能である．

□ PDFを直接生成する方式

図におけるDocument→Viewerの流れである．中間ファイルであるPostScript生成，後処理のDistillerを経由しないため，高速にPDF生成を行うことができる．現在は極一部のAdobe製品にしかこの方式は実装されていない．しかしシステム開発者のためにAdobe PDF Libraryというライブラリ製品が提供される予定である（1998年3月現在）．

おわりに

電子出版の一手法としてすでに利用されているPDFのデータとしての特性に焦点を当てて述べた．AcrobatはPDFを閲覧するための製品であるが，Acrobatが提供する機能がPDFのすべてではない．PDFは現状のAcrobat本体が提供する機能以外の機能を付加するための拡張性や他システムとの融合性など非常に幅広く活用することができる．既存の概念を超えた次世代の「電子出版」に対応する可能性を秘めている．

Acrobat SDKに関する情報入手先：
<http://www.adobe.com/supportservice/devrelations/sdks.html>

このページからAdobe Acrobat SDKを辿る．本編で述べているPDFの詳細はさらにDocumentation and SpecificationsのPortable Document Format Reference Manual Version1.2（Includes FDF specification）（1.5MB）を参照のこと．

日本国内におけるサポートサービスに関しては
<http://www.adobe.co.jp/support/developer.html>の「Adobe Acrobatプログラム」を参照のこと．

（平成10年3月31日受付）