Guide for PDF Developers
全てのＰＤＦ開発者のために

第二版

# ＰＤＦ解析ガイド

第二版

*How to Analyze PDF*

**KiyoriSystem**
キヨリシステム

**Kiyori Minoura**
**箕浦　喜順　著**

# はじめに

多くの人は、ＰＤＦファイルをワープロ形式の一つとして認識しているでしょう。

しかし、ＰＤＦの本来の形は、一種の単なるアーカイブファイルであり、Zip や Lha の仲間です。ＰＤＦの基本的なデータ構造は、インデックス情報とその実データの集まりです。

実データには、データ毎に各種のエンコードやデータ圧縮、暗号化を施すことができ、かつ、その変更履歴を管理、保管することができます。この点が、他のアーカイブツールと一線を引くものとなっています。

データ記述には、文字型や整数、配列型、辞書型、ストリーム型などが定義されていて、それらを系統的、かつ階層的に使うことができます。その内容には、コンテンツストリームを呼ばれるベクトルデータやタイポグラフィー、フォントデータが含まれているので、その結果ワープロデータ的な使われ方が主流になっています。

加えて、AcrobatReader などが、ワープロデータ的なベクトルデータのレンダリング機能を備えており、文書をビジュアルに表現します。

このように、ＰＤＦは、特定の用途のエンコードや暗号化、データ構造を持つことなく、汎用的な思想のもとに設計されているにもかかわらず、ワープロ形式の一つとして認識されてしまっているのは残念です。しかし、ワープロ形式に付随して、フォント埋め込みや添付ファイル、署名情報などが追記できるのは他の形式にはない利点となっています。

ＰＤＦの基本的な構造は、そのバージョンが変わってもまったく変化せず、それが汎用的であったことを想像させますが、ビューアを選ぶような閉塞的な形式は次第にＸＭＬなどの形式に取って変わられていくでしょう。しかし、ＰＤＦの持っている思想は、形が変わっても引き継がれていくはずですので、きっとＰＤＦの構造を理解するのは今後の役に立つはずです。

ＰＤＦ内部のデータ内容は、エンコードや圧縮、暗号化の影響を受けるので、テキストエディタなどでは直接参照できないことがあります。そうなるとテキストエディタやバイナリーエディタでは解析は不可能です。本書では、添付の解析ツールを使いＰＤＦの内部構造を理解します。

内部構造を理解し、色々なＰＤＦに触れると、ＰＤＦに対して色々なことが可能になりますし、ＰＤＦの思想を再認識し、新たなＰＤＦソフトウェアやその思想を受け継ぐものを開発する手助けになるでしょう。

最後に、本書は、ＰＤＦの全体的な構造を解説するもので、パスワードの解析やセキュリティ情報の解除を目的としたものではありません。誤解の無いようにお願いします。

また、本書は、解析ツールを使いこなすための予備知識を説明するためのもので、ＰＤＦの詳細な仕様に関しては、ＰＤＦリファレンス等を参照してください。



箕浦喜順
E-mail:kiyori@kiyori.co.jp

# 目　　次

# 1. PDF の基本構造

## 1.1. 基本構造

基本的に、ＰＤＦは、インデックスデータとそのデータ本体から成り立ちます。
ＰＤＦは、テキストエディタなどで見てみると、数字の連番が並んだところと、文字が化けて読めない部分が見てとれます。
最初 Header と Trailer と呼ばれる部分から、解析し始めて、次に xref と呼ばれるインデックス情報を取得します。
数字が連番になって３つの値が並んでいるところが xref というインデックス情報になります。
この情報をもとに、データ本体を取得します。
最初の数字がオブジェクト番号で、次が履歴番号、ｆやｎは有効、無効の意味です。

データ本体は、xref が指し示す先に存在します。
オブジェクト番号、履歴番号で開始し、
実データが続きます。

```
xref
1 1
0000010138 00000 n
3 1
0000010214 00000 n
19 1
0000010367 00000 n
22 1
```

xref データの例

```
1 0 obj<</Pages 2 0 R/Type/Catalog/AcroForm 15 0 R/Metadata 29 0 R>>
%PDF-1.5
%粵ﾏﾓ
1 0 obj<</Pages 2 0 R/Type/Catalog/AcroForm 15 0 R/Metadata 29 0 R>>
endobj
2 0 obj<</Count 1/Kids[5 0 R]/Type/Pages>>
endobj
3  0  obj<</ModDate(D:20030826143941+09'00')/CreationDate(D:20030826014040+09'00')/Creator(Adobe Illustrator 10)/Producer(Adobe PDF library 5.00)>>
endobj
```

ＰＤＦオブジェクトの例

このようにある程度は、テキストエディタでもその構造を追うことができますが、ちょっとしたＰＤＦは、オブジェクト構造も履歴などが複雑に入り組んでいて、バイナリーデータも随所に混じるのでとても目で追えたものではありません。
ＰＤＦ関係のソフトウェアの開発者であれば、その苦労は理解してもらえると思います。
本解析ツールでは、こういった XRef 情報を意識して参照する煩わしさは一切ありません。

# 1.2.データ形式

ＰＤＦ上の基本データには、いくつかの種類があります。ここでは、ＰＤＦを解析するのに必要となる基礎知識のみを記します。より詳しく知りたい場合は、ＰＤＦリファレンスを参考にしてください。

データ内容は、エンコードや圧縮、暗号化の影響を受けるので、テキストエディタなどでは直接参照できないことがあります。そうなるとテキストエディタやバイナリーエディタでは解析は不可能ですので、付属のツールで雰囲気を掴んでください。

（１） ブーリアン型
true,false でのみ表現される型です。

（２） 数値型
数値を表現します。ＰＤＦ上の多くの数値情報は、実数値です。

（３） 文字型
()で囲まれ、文字列を表現します。<>で囲まれると１６進表記になり、ユニコードで表現します。
文字で表現される日付型やテキスト型もこの形式になります。テキスト型には、特殊なエスケープシーケンスが混じることがあります。

（４） 名前型
文字型に似ていますが、/で必ず始まります。特定の名称や辞書型のキーなどに使われます。

（５） 配列型
データの羅列だけを複数持ちます。[]で囲まれています。
配列の中には、さらに配列や辞書などの他のデータ型や参照型を持つことがほとんどです。

（６） 辞書型
キーとデータの組みを複数持ちます。<<>>で囲まれています。
ほとんどの辞書は、Type エントリを持ち、どうゆう辞書なのかを知ることができます。
また、SubType エントリを持つ辞書は、さらにその２次的な属性を知ることができます。
辞書の中には、さらに辞書や配列などの他のデータ型や参照型を持つことがほとんどです。
非常に多くの辞書型のデータがあります。

（７） ストリーム型（XObject）
辞書型を拡張した型で、さらにストリームデータを持っています。XObject 型とも表現します。
stream～endstream までがそのストリーム内容になります。ストリームがどんなデータなのかは、辞書の中に記述されます。
Filter パラメータを持っており、エンコードの方式を知ることができます。
ページ内のベクトルデータや添付ファイルのデータ内容などは、ストリームで格納されます。

（８） 参照型
上記の参照アドレスのみを持っています。”５０R”のような３つのデータの組み合わせになります。

ＰＤＦ上のデータは、以上のデータ型が、お互いに複雑に入り組んで表現されます。

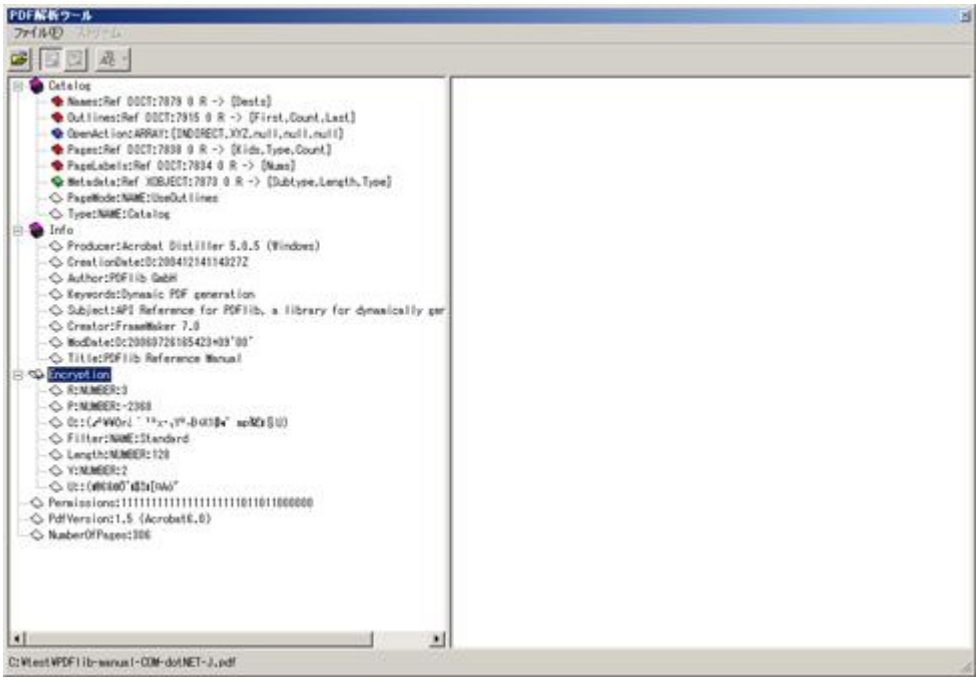
# 2. ＰＤＦの解析方法

## 2.1.解析ツールの使い方

解析ツールでＰＤＦを読む込むと、まず左側のペインに、ツリー構造が表示されます。ストリーム内容が存在する場合は、右ペインにストリーム内容が表示されます。

このツリーは、カタログツリーというもので、ＰＤＦ内部のデータ構造をツリー構造で表現しています。

このツールでは、テキストエディタで見ると直線的だった PDF の内部構造データを、Catalog 辞書をルートとするツリー構造で系統的に把握することができます。




各ノードの表示形式は、

**表示形式**

**オブジェクト名：データ型：データ値**

※　型が参照型の場合には、Ref を頭につけた参照先の型を表示

**ref データ型:アドレス　-> 参照先の内容**

となっていて、型が参照型の場合には、参照先データのデータ型の前に Ref を付けて表示します。

参照先が単純な型の場合は、その内容も合わせて表示します。

データが、参照型の場合は、クリックでその参照先を子階層として展開します。

また、辞書型、配列型の場合は、値として、その内容の一覧が展開されます。

辞書および配列オブジェクトの場合は、ダブルクリックで都度、子階層として展開表示します。

それぞれのエントリには、parent,page などの上の階層を指すものや、全く別の階層にジャンプするものも含まれています。それで、クリックにより都度階層を展開する方法を取っています。

本来のＰＤＦの内部構造は、明確なツリー状になっているわけでなく、複雑な蜘蛛の巣状になっていると考えてください。しかし、Catalog をルートとするツリー構造で理解すると大変わかりやすく構造を把握することができます。

## 2.2. カタログとは

ＰＤＦに保存されているデータは、ほとんどカタログツリーの配下に存在します。
カタログツリーとは、ＰＤＦのルートディレクトリのようなものです。
たとえば、ページ内容を表示するためのページコンテンツは、Catalog¥Pages 配下に存在し、ページの構成情報もこのツリーに存在します。
フォーム情報は、AcroForm 配下に存在します。
名前付き宛先などの情報は Names 配下に、JavaScript などの記述は、JavaScripts 配下に存在します。

## 2.3. カタログ外のデータ

カタログは、ルートディレクトリみたいなものですが、カタログ以外にもルートとなる情報が存在します。
これらの情報は、カタログ情報にたどり着くための先行情報みたいなものです。

（1） ＰＤＦヘッダー
ヘッダには、ＰＤＦファイルのバージョンが記述されます。

（2） 暗号化辞書
ＰＤＦの暗号化に関する情報を持っています。
すべてのＰＤＦデータは、同じ方法で暗号化されています。
最初にデータを読み込む時から暗号化辞書は必要となります。

（3） セキュリティパーミッション
また、Acrobat で色々な動作が制限されるのは、このパーミッション情報を元に制限しているからです。

（4） 文書情報(Info)辞書
ＰＤＦの作成ソフトや作者や更新日時などの情報です。

# 2.4. ＰＤＦオブジェクト

カタログ内の各オブジェクトには、暗黙のルールがいくつかあります。
それを理解すると、より速くＰＤＦの構造に親しむことができます。

まず、各オブジェクトには、概ね判りやすい名前が付いています。
辞書型の場合は、Name パラメータ、Type パラメータを持つことが多く、その辞書が何の辞書であるかわかるようになっています。辞書型には、非常に多くの種類の辞書があります。
さらに、Type パラメータだけで分類できない場合、Subtype パラメータを持つことがあります。

多くのＰＤＦオブジェクトは、お互いに参照し合って存在します。
まるで、出来の悪いＷＥＢページのような構造をしています。
Kid 配列は、そのオブジェクトに子オブジェクトが存在することを指し、それらの参照を配列で持ちます。
この逆で、Parent パラメータは、親オブジェクトを指します。
また、P パラメータは、通常ページを指す事が多く、該当するページ辞書への参照を指します。

個々の辞書には、配下に Resource 辞書を持つことがあります。これは、そのオブジェクトが単独で利用する外部データのようなものです。中には、フォント、イメージ画像、カラースペース、埋め込みファイルなどが入ることがあります。

また、NULL オブジェクト “００Ｒ”が時には使われることがあります。このデータに意味は有りませんが、ダミーのエントリやオブジェクトを使いたいときに利用されます。

PDF オブジェクト内の文字データは、基本的に Unicode で記述されます。
しかし、中には例外もあります。FileSpec 辞書は、そのＰＤＦが生成されたＯＳに依存する傾向があり、ファイル名などにそのＯＳ固有の文字情報を持ちます。日本語 Windows 上で埋め込みファイルが行われたＰＤＦは、そのファイル名などがＯＳ固有の SJIS コードで記述されることがあります。

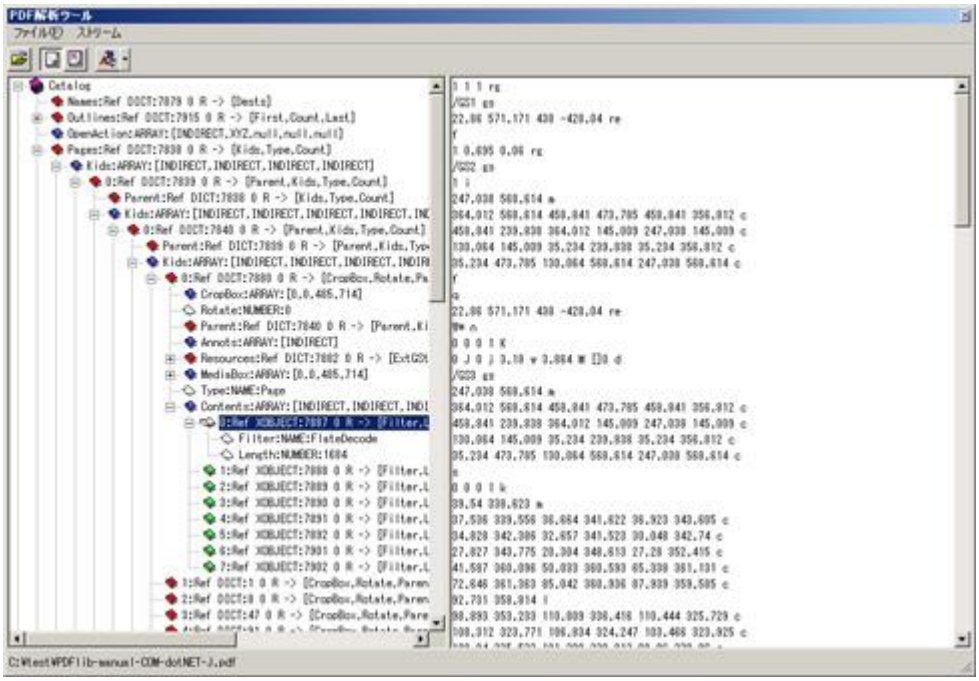
また、栞のタイトルなどの一部の特殊なテキスト情報には、マルチ言語化されたテキストが入ることがあります。この場合、PDF 独自の ESC シーケンスで国コードや言語コードが割り込みます。
通常の文字エンコードでは、文字化けして正しく表示ができませんので、ＨＥＸモードで確認してください。

# 3. いろんなケース

## 3.1. 各ページのコンテンツストリームを見る

ページオブジェクトは、カタログツリーの Pages から手繰っていきます。
ページそのものは、段落構造に合わせてさらに階層構造を持っています。この親子関係は、Kid,Parent の組み合わせで、成り立っています。Kids 配列があれば、さらにその先の子を深追いしていきます。
そして、最終的にページ辞書が現れます。
各ページ辞書の出現順が、結果的にページ数の順番になっています。

ページ辞書を見ると、そのページに関連した情報がさらにその配下に現れます。
ページのベクトルグラフィックスであるコンテンツストリームは、このページ辞書の Contents 配列に含まれるストリーム型データのストリーム部分に保存されます。ここで、ストリームを開くとコンテンツの内容を確認できます。
コンテンツストリームはベクトルデータなので、一見チンプンカンプンな文字列が並んでいます。
これらの意味は、ＰＤＦのリファレンスを見てください。注意するのは、基本的にＰＤＦの座標は、左下が原点になっていることです。WindowsAPI などに慣れていると左上を基点に考えてしまいますので、そのまま認識すると上下が反転してしまいます。もちろん、座標変換や回転などがパラメータで指定されていると、さらに座標の位置づけが変わってきますので注意が必要です。



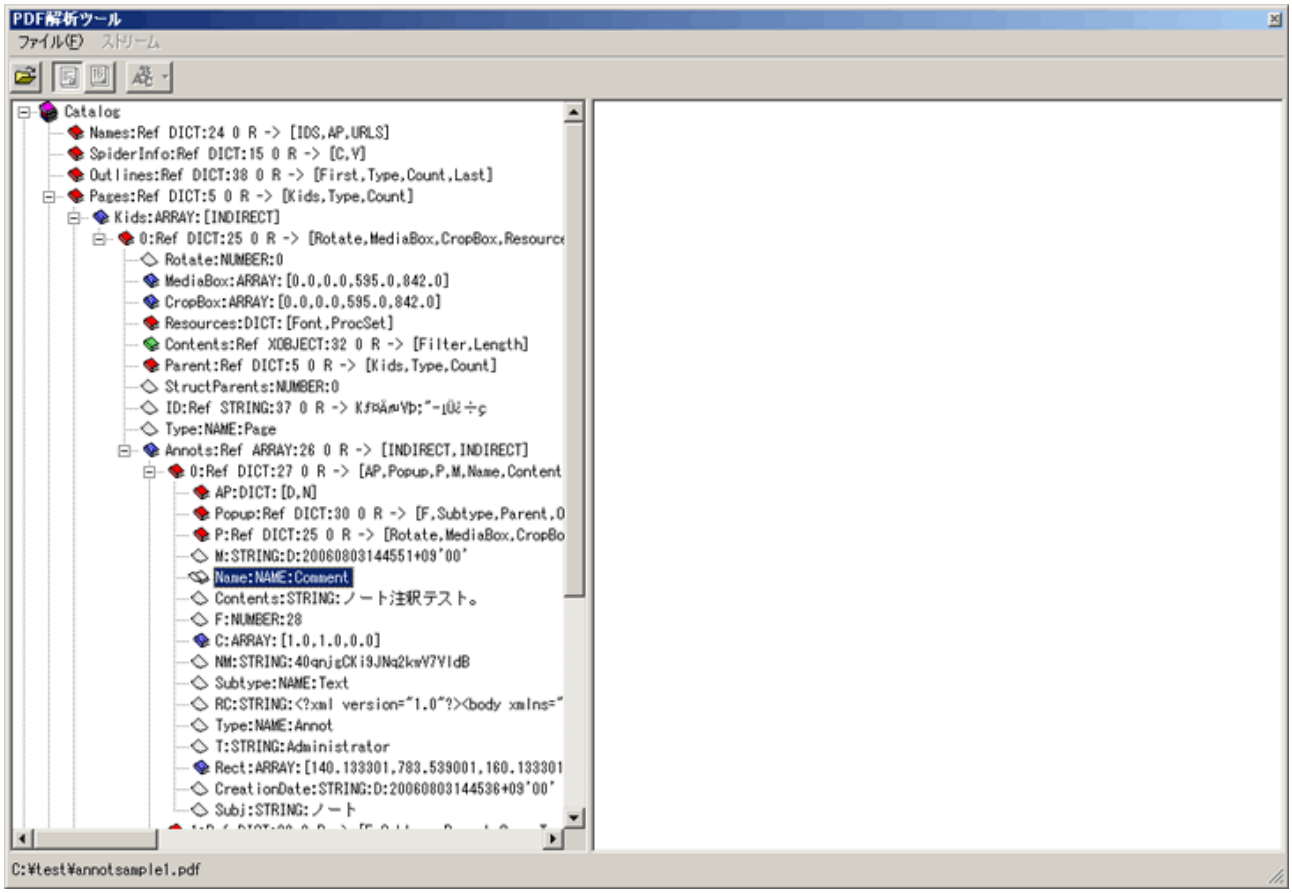

ツリー上で、ストリーム型(XOBJECT と表現)を選択すると、右ペインでは、このようにその内容をテキスト表示します。ＰＤＦ内のテキストエンコードは、通常 Unicode ですが、まれに、Unicode 以外のエンコードも使われます。

# 3.2. 注釈をみる

ＰＤＦには、注釈を後から追加することができます。ＰＤＦの本体は、ベクトルデータのカタマリなので、それを自由に編集するのは容易ではありません。そういった不便さを解消するために、後付でなんらかの修正やコメントを加えるために用意された機能が注釈です。本文そのものを修正するだけの機能は有りませんが、訂正やコメントを追加したり、場合によっては、ページ全体を覆って全面的にページ内容を差し替えることも可能です。注釈には、多くの種類が用意されています。

注釈には、動画や音声、アニメーションなどの動的なコンテンツを埋め込むことができます。
また、スタンプやアイコンなどのビジュアルな情報が注釈として添付されていることもあります。もちろん、コメントそのものも解析対象になることがあるでしょう。

注釈データは、ページ毎のページオブジェクトの Annots 配列に登録されます。
個々の注釈は、Annots 配列の中の Annot 辞書として定義されます。
Subtype 属性により、その注釈のタイプを判定でき、注釈の種類別に、他の属性も異なります。
注釈には、外観（見た目）が必ず存在し、注釈辞書の AP 辞書で定義されます。
通常の注釈の外観は、このＡＰ辞書のＮ（Normal）外観で定義されます。このＮ外観は、ストリーム型になっていて、そのコンテンツストリームにベクトルデータが保管されます。
また、注釈がクリックされた場合には、A 辞書によりアクションが実行されたり、AA 辞書によりイベントマップが定義されたりします。



# 3.3. 添付ファイルを開く

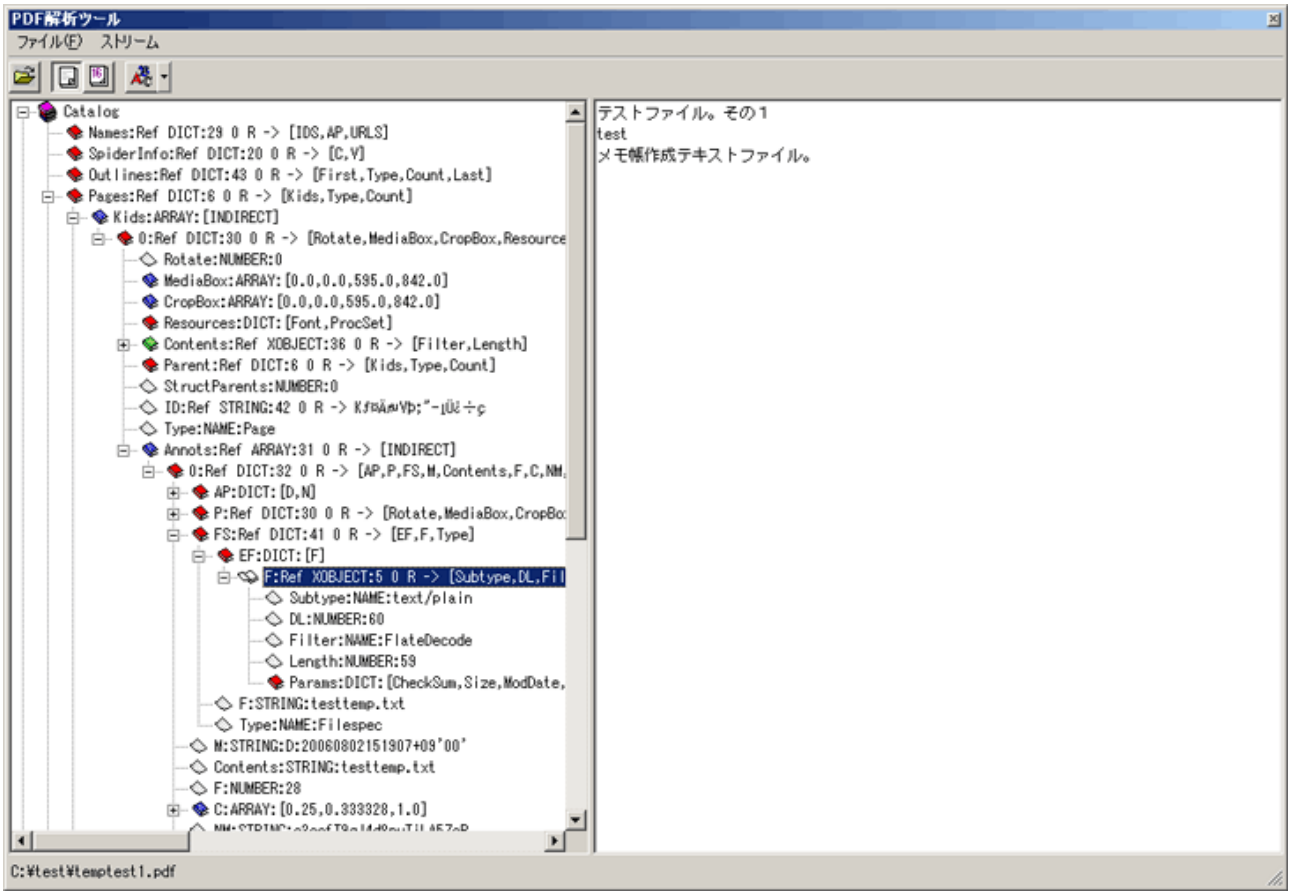
ＰＤＦには、ファイルを添付することができます。
ＰＤＦのファイル添付方法には、歴史的な経緯で、２つの方法が定義されています。

（１） 埋め込みファイル（EmbbedFile）
Catalog/Names/EmbeddedFiles の配下にファイルストリーム辞書のエントリを追加することで生成します。
埋め込みファイルは、注釈ではないので、アイコンなどの外観イメージは存在しません。

（２） 添付ファイル注釈（FileAttach Annotation）
注釈の一種で、アイコン（虫ピンなど）やメモと共に、ファイルを添付することができる注釈です。
注釈辞書のエントリに、ファイルストリームを格納できるようになっています。
注釈ですので、ページ単位に存在する Annots 注釈配列に含まれます。

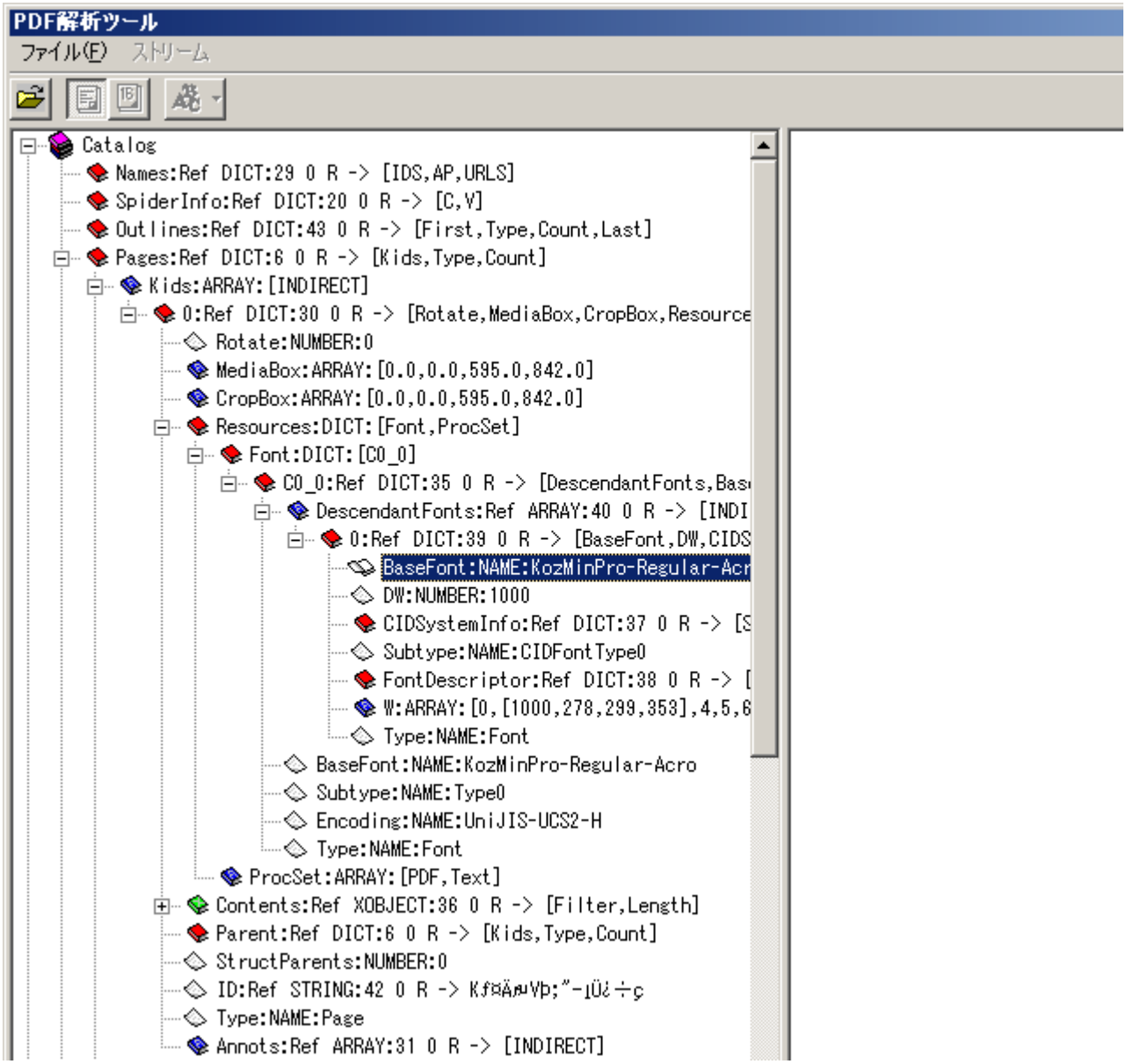
Acrobat では、このどちらも利用することができます。ユーザは、このどちらかを好んで使うことになります。どちらの場合も、ファイルストリーム辞書という共通の形式で記録されます。
ファイルストリーム辞書は、ストリーム型で、そのストリーム部分にファイル本体が記録されます。

プレーンテキストファイルを添付した場合は以下のように表示されます。



# 3.4. フォント情報を見る。

PDF には、フォントを埋め込んで配布することができます。
フォントはデジタルデータとしてＰＤＦ内に埋め込まれます。
この事で、実行環境などの影響を抑えることができ、国際化された文書や、デザイン的に優れた形で配布することが可能です。
フォントは、必要な文字の分だけ、フォントリソースという形で埋め込まれます。
コンテンツストリーム上は、このフォントリソースを参照することで、フォントのエンコーディングを行います。
フォントリソースは、まとまって存在するのはではなく、ページ辞書や注釈のそれぞれコンテンツが必要とする分だけ、それぞれの配下に存在する Resource 辞書に含まれています。

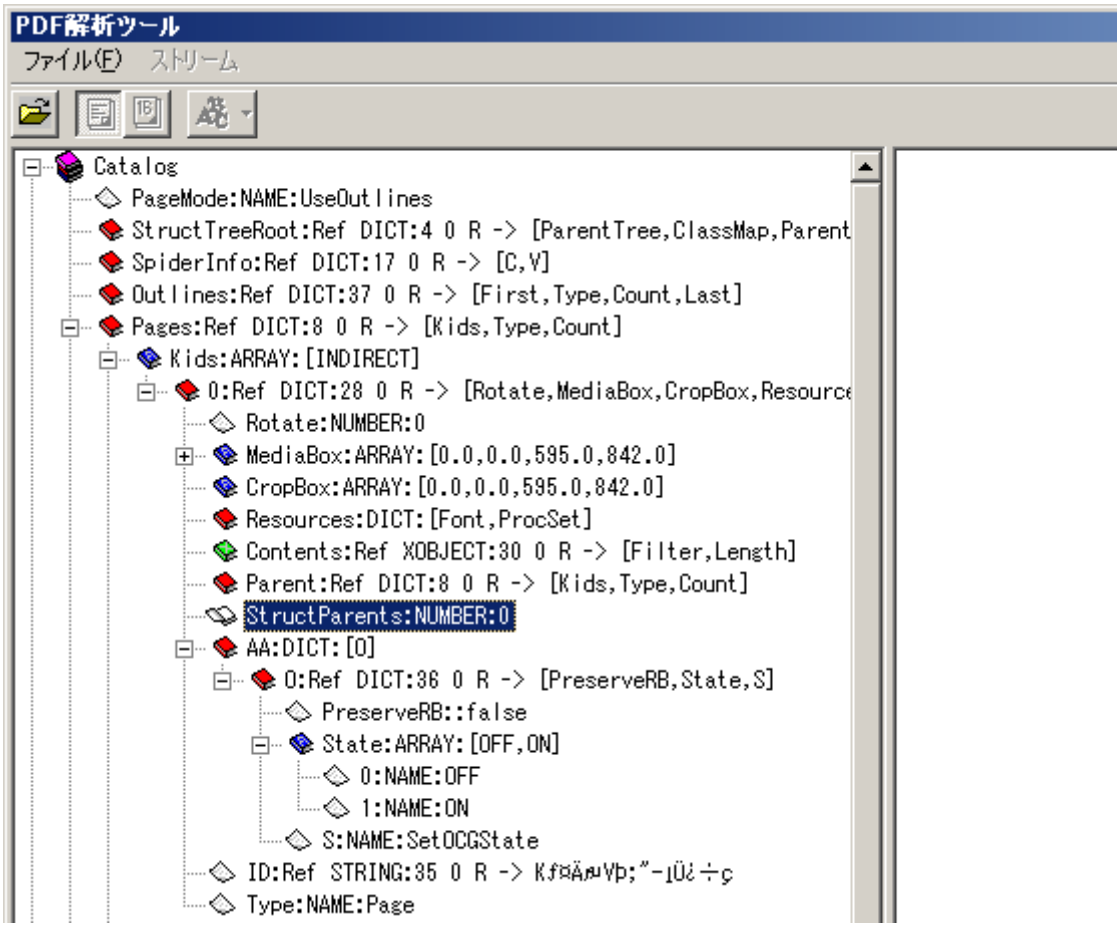


# 3.5. Action の流れを知る

ＰＤＦ上には、Action という対話的な機能を組み込むことができます。
この機能により、ＰＤＦは静的なものから動的なものへと変化します。
Action には多くの種類があります。代表的なものは Goto アクションで、ページを移動したり、指定された URL を表示したりします。他にも、記述された JavaScript を実行したり、他のプログラムを起動したり、指定ページの印刷を行ったりすることもできます。
どのようなアクションが使われ、記述されているかは、ＰＤＦを直接参照することで知ることができます。
アクションは、以下のような対話的な場面で、随時実行されます。

（１） OpenAction
Catalog/OpenAction 辞書で記述されて、文書のロード時に実行されます。
（２） ページアクション
ページを表示したり、印刷したりするときに、指定されたアクションを実行します。
（３） イベントアクション
リンク注釈や、フォームのボタンをクリックすると指定されたアクションを実行します。
フォームには、より詳細なイベントアクションが指定できます。
（４） 栞アクション
栞のエントリをクリックすると、指定されたアクションを実行します。

それぞれの機能が該当する場所には、Action 辞書または、ＡＡ辞書（イベントマップ）が記述されています。古いＰＤＦでは、Goto アクションの代わりに、Dest 記述がされていることがあります。



## 3.6. 栞を知る

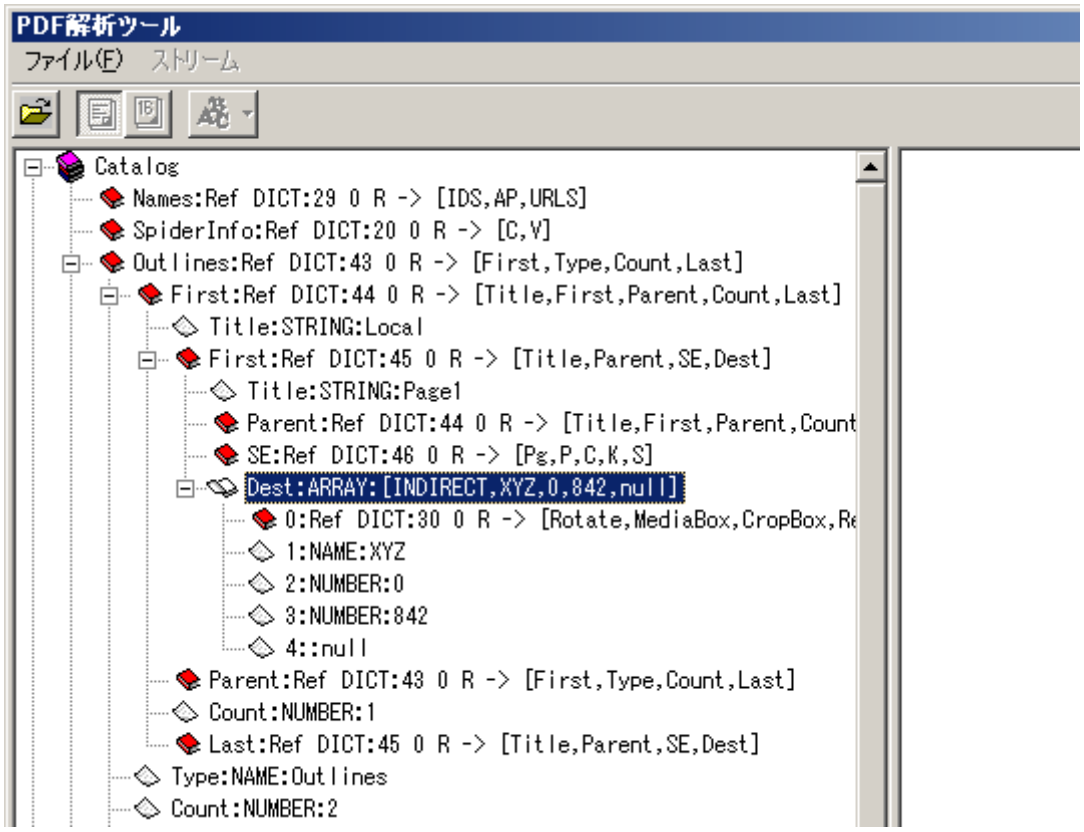
Catalog/Outlines 配下には、ＰＤＦの栞情報が記録されています。栞データは、再帰的な構造になっていています。

栞の一行分のデータは、Outline 辞書という共通の形式で記述され、それが再帰的に構成されます。このため、必ず Kid 配列や parent というパラメータを持っており、また、栞の順序を示す、First,Last,Next,Prev といったリンク情報を持っています。

また、栞の機能として、Dest 飛び先や、Goto アクションといったアクション辞書を必ず持っています。

飛び先には、直接ページオブジェクトなどを指す以外に、名前付き飛び先（Named Destination）というものがあります。その場合、Dest の飛び先には名前だけが指定されて、その実態は、名前辞書に記述されます。

名前辞書は、Catalog/Names/Dests に再帰的に記述されます。その場合の名前には、“.”を区切り文字として階層構造を表します。そして、Kids 配列を追っていき、最終的に辿り着くオブジェクトが名前付き飛び先の目的地になります。



# 3.7. JavaScript を解析する。

ＰＤＦに記録されている JavaScript を覗きたいケースが多々有ります。
それは、Acrobat の JavaScript ではバージョン間の互換性が低く、まれに問題になり、詳しく調べたいときがあるからです。
しかし、Acrobat などのツールは、JavaScript にあまり親切ではありません。
だから、直接ＰＤＦ上の JavaScript を参照し、どのような構成になっているかを把握する必要があります。

Acrobat の JavaScript は、結構複雑です。
JavaScript が一箇所にまとめて記述されるとは限らず、各所に分散しているからです。
すべての JavaScript をツリーから見つけ出すのは意外と厄介ですので、この場合は、エディターなどで検索して、記述箇所にあたりをつけたほうが早いかもしれません。
Action には、Action 辞書が直接記述される場合と、AA 辞書によりイベントマップが記述される場合があります。

（１） JavaScript 辞書
Catalog/Names/JavaScripts には､文書毎にインプリメントされる JavaScript が記録されています。
これらの JavaScript は、文書のロード時に実行されます。

（２） OpenAction 辞書
Catalog/OpenAction により、文書のロード時に実行されるアクションが記述されます。

（３） Page 辞書のＡ，ＡＡ辞書
Page 辞書には、オープンアクション、印刷アクションなどが記述されることがあります。
該当するページが表示されたときに、実行されるアクションのことで、この時点で JavaScript を実行することがあります。

（４） 各種イベントの Action 辞書
フォームのボタンクリックや、各種イベント（フォーカスイン、フォーカスアウト）などはアクションが割りつくことがあります。当然、アクションの中で、JavaScript を実行することがあります。
アクションが記述されるのは、フォームだけとは限りません､栞のクリックイベントなどで自由なアクションの記述が可能です。

JavaScript そのものに関しては、JavaScript のリファレンス AcroJS70.pdf を参照してください。

# 3.8.イメージ画像を見る。

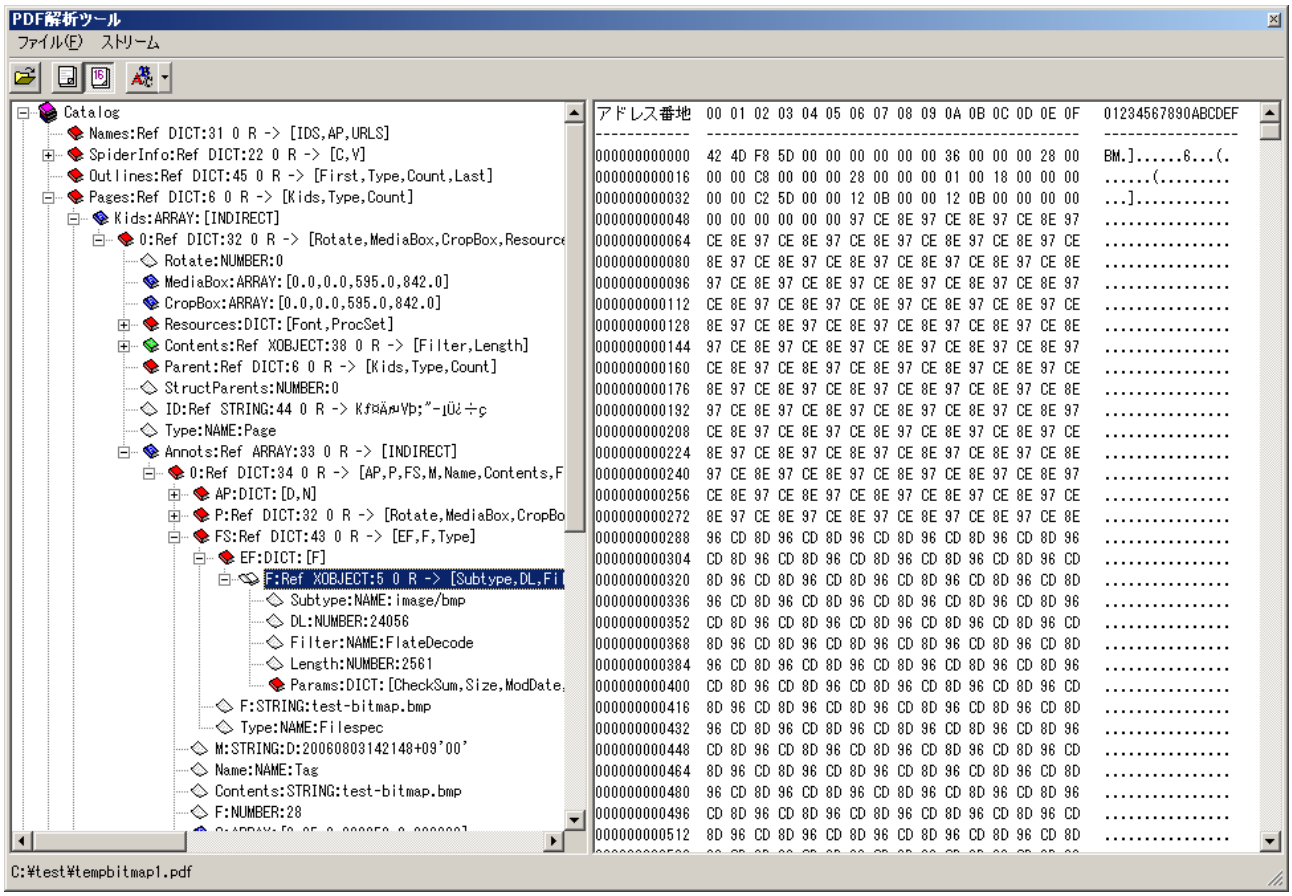
コンテンツストリーム上では、イメージ画像は直接記述されていません。これは、ページリソースや外観リソースに配置され、その参照情報としてコンテンツ内で記述されます。
ページ辞書には、Resouces というキーが存在することがあります。
その中に XObjects という辞書が見つかれば、それが画像データの辞書です。
中には、画像データがストリーム型として登録されているはずです。
個々のデータ形式は、その画像データの辞書を見てください。ビット数やサイズ、エンコード方法などを知ることができます。そのストリーム内に、画像データそのものが記録されています。

イメージデータの保存場所：ページオブジェクト/Resources/XObjects/

なお、イメージデータのストリームには、以下のエンコードがされることがあります。

| |
|---|
| ASCIIHexDecode、ASCII85Decode、LZWDecode、<br>Flate、RunLengthDecode、CCITTFaxDecode、DCTDecode |

本ツールでは、Flate エンコードのみに対応しています。その他の形式でエンコードがされている場合は、ストリームをＲＡＷデータで出力保存し、各自でエンコード処理を行って下さい。



# 3.9. フォーム（署名）を見る

フォームの各要素は、注釈の一種で Widget 注釈と呼ばれるものです。ですから、注釈の章で説明した Annots 辞書にその実態が記録されています。この事から、フォーム要素は、ページに関連したオブジェクトと捉えることができます。

しかし、それとは別にフォームの場合には、その要素を文書全体で順序立って持つ必要があります。

カタログ直下にある AcroForm 辞書がそれにあたります。

フォームの要素は、このようにページに関連した注釈配列の一部として存在すると同時に、Catalog/AcroForm の Fields 配列の要素としても存在します。

フォームのフィールドには、テキストボックスや、ボタンのようなコントロールに近いもの以外にも、不可視フィールドや、署名情報などが存在します。これらの情報は、通常 Acrobat などでは、直接アクセスすることはできません。不可視な情報に、なにがどのように書かれているかは、解析ツールで覗いてみて初めて判ります。

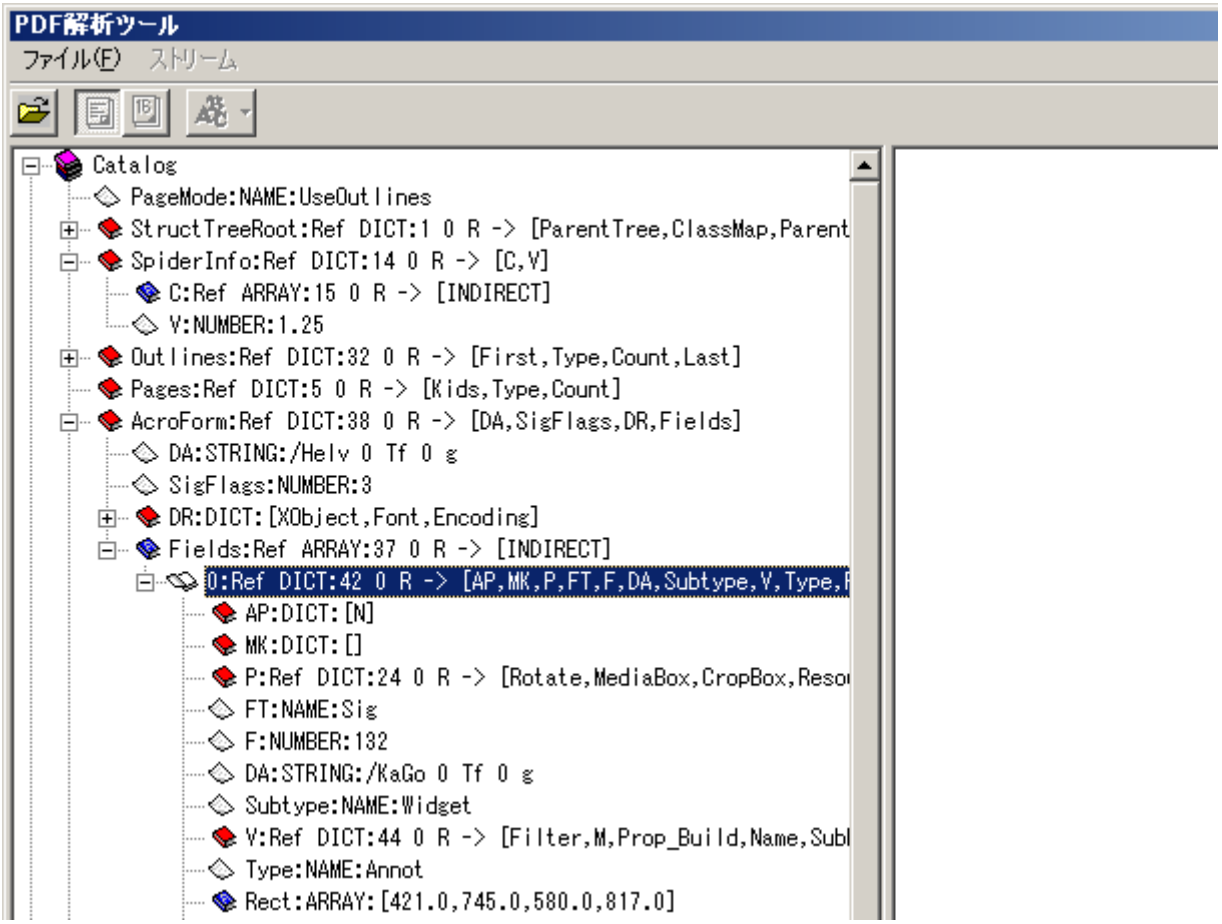
Fields 配列は、さらに Kids 配列を持つことがあります。これは、構造化フォームを表し、フォームの要素が階層的に構成されていることを指します。Kids 配列は、さらに Kids 配列を持つことができ、より深い階層構造を持つことがあります。ラジオボタンなどのフィールドは、最初からグループ化されています。

フィールドの値は、通常 v 要素がその持っています。また、その値に対応する外観をＡＰエントリのストリームが持つことになります。

フィールドが階層構造を持つ場合、フィールド名には、完全フィールド名が割り当てられます。それは、“.”を区切り文字とするパス名のようなもので、フィールドに直接アクセスような場合には、この名前が使われます。

また、フィールドの種類は、フィールド辞書のＦＴパラメータを見ると判断できます。

署名フィールドは、この値が/Sig になっていて、Ｖパラメータに署名辞書が作成されます。



# 3.10.メタデータを見る。

ＰＤＦの文書情報には、ＰＤＦで規定されたものだけでなく、ユーザ固有の情報を持たせることができます。そういったＰＤＦで規定された以外の情報は、メタデータとしてＰＤＦ内に記録されます。

メタデータは、Catalog/Metadata にストリーム型として保持され、そのストリームデータ上にＸＭＬの形式でデータを保管します。