PDF 规范
第六版

Adobe®便携式文件格式

版本 1。7

2006 年 11 月

Adobe 系统公司

目录

前言

第 1 章 介绍

1.1 关于本书

本书提供 PDF 文件格式的描述,主要面向直接创建 PDF 文件的 PDF 制作应用程序的开发人员. 它还包含足够的信息，允许开发人员编写读取现有 PDF 文件并解释或修改其内容的 PDF 消费者应用程序。

虽然 PDF 规范与任何特定的软件实现无关,但通过描述典型应用程序处理方式,可以最好地解释一些 PDF 功能。在这种情况下，本书使用 Acrobat 系列 PDF 查看器应用程序作为其模型。（原型查看器是完全功能的 Acrobat 产品,而不是 AdobeReader®产品）。附录 C 讨论了 Acrobat 查看器应用程序中的一些实施限制，但这些限制并不是文件格式本身的一部分。附录 H 提供了兼容性和实现说明,描述了 Acrobat 查看器在遇到不了解的较新功能时的行为方式,并指定了

Acrobat 产品与本书中提供的规范不一致的领域。PDF 制作者和消费者应用程序的实施者可以使用此信息作为指导.

此版本的 PDF 规范介绍了 1.7 版本的 PDF（见附录 H 中的实施说明 1）。在整本书中，PDF 的特定版本的特定信息标有(PDF 1.3）或（PDF 1.4）等指标。在该版本中，如此标记的特征可能是新的或基本上重新定义的.指定的功能（PDF 1。0）一般在以后的版本中被取代；除非另有说明,否则特定于其他版本的特征也被理解为在后续版本中可用。（为特定 PDF 版本设计的 PDF 消费者应用程序通常忽略他们不认识的较新功能;附录 H 中的实现说明指出了异常。）

注意：在本版本中，消费者一词通常用于参考 PDF 处理应用程序；查看器保留用于实现与用户交互的功能的应用程序。然而，这种区别并不总是很清楚，因为非交互式应用程序可能会处理表示交互功能的 PDF 文档（例如注释）中的对象。

本书的其余部分如下：

- 第 2 章“概述”简要介绍了 PDF 的整体架构及其背后的设计注意事项，并将其与 PostScript 语言进行比较，并描述了它们共享的基础图像模型.
- 第 3 章“语法”介绍了 PDF 在对象，文件和文档级别的语法。它为后续章节提供了一个基础，它们描述了该信息如何被解释为页面描述，交互式导航辅助工具和应用程序级逻辑结构。
- 第 4 章“图形”描述了用于描述 PDF 文档中页面外观的图形运算符。
- 第 5 章“文本”讨论了 PDF 的特殊设施，用于以字体形式呈现文本，或者由字体定义的字形。
- 第 6 章“渲染"考虑了与设备无关的内容描述如何与特定输出设备的特性相匹配.
- 第 7 章“透明度”讨论了在 PDF 1.4 中介绍的透明成像模型的操作，其中可以以不同程度的不透明度绘制对象，允许页面的原内容显示。
- 第 8 章“交互功能”介绍了 PDF 的功能，允许用户使用鼠标和键盘与屏幕上的文档交互。
- 第 9 章“多媒体功能"介绍了支持嵌入和播放多媒体内容(包括视频，音乐和 3D 图形)的 PDF 功能。
- 第 10 章“文档交换"显示了 PDF 文档如何合并可用于应用程序之间的文档交换的更高级别的

信息。

- 附录 A“运算符摘要”列出了用于描述 PDF 文档的可视内容的所有操作符.
- 附录 B“类型 4 函数运算符”总结了可在 PostScript 计算器函数中使用的 PostScript 运算符，该函数包含用 PostScript 语言的一小部分编写的代码。
- 附录 C“实施限制”描述了 Acrobat 查看器应用程序施加的典型大小和数量限制。
- 附录 D“字符集和编码"列出了假定在任何 PDF 消费者应用程序中预定义的字符集和编码。
- 附录 E“PDF 名称注册表”讨论了由 Adobe Systems 为开发人员维护的注册表，其中包含 PDF 生成器或 Acrobat 插件扩展使用的私有名称和格式。
- 附录 F“线性化 PDF”描述了一种特殊形式的 PDF 文件组织，旨在在网络环境中高效工作。
- 附录 G“PDF 文件示例”介绍了几个示例，显示了实际 PDF 文件的结构，范围从包含最小单页文档的结果到显示 PDF 文件结构在多个修订版本中演变的结构。
- 附录 H“兼容性和实现说明”提供了 Acrobat 浏览器应用程序行为的详细信息,并描述了消费者应用程序如何处理包含他们无法识别的功能的 PDF 文件。
- 附录 I“对象杂凑的计算”详细描述了一种用于计算对象杂凑的算法（在第 8。7 节“数字签名”中讨论过）.

彩色板部分提供了一些 PDF 与颜色相关的特性的插图。在“见第 1 版”的文本中引用本节的内容。

本书的结尾是参考文献和索引。

1.2 PDF 1.7 功能简介

在 PDF 1。7 中介绍或修改了一些特性。下面列出了最重要的添加内容，以及对这些添加内容的主要部分的引用：

1.2.1 3D 图形

PDF 1。7 引入了新功能,增加了 PDF 查看应用程序对 3D 图形外观和行为的控制：

- 更多地控制 3D 图形的外观，而无需更改原始图稿，也不需要使用嵌入式 JavaScript.3D 图形的具体视图可以指定图形应如何呈现,着色，点亮和横截面.他们还可以指定应在视图中包含 3D 图形的哪些节点（三维区域），哪些节点应放置在视图中，以及它们是否应该是透明的。这些特征可以显现出难以查看的几何区域。
- 在 3D 图形的特定视图上放置标记注释的功能。这可以确保适用于 3D 图形的标记随后可以相对于艺术作品整体和艺术作品中的各个元素而正确显示。 应用于 3D 图形的标记注释提供了一种确保从应用标记注释开始的图形没有更改的方法。
- 控制 3D 图形激活时显示的用户界面和工具栏.
- 控制关键帧动画的时间框架，重复和播放风格。游戏风格是线性重复（如步行角色）和基于余弦的重复(如爆炸式收缩图像）。

1.2.2 交互功能

标记注释的一些附加功能使它们更适合于技术通信和审查,或者用于法律环境.

协助技术交流的交互功能

标记注释的几个补充有助于技术沟通和审查：

- 添加多义线和多边形标注注释的维度意图。维度意图支持用户提供的维度信息与构成折线和多边形标注注释的线段的关联。此功能与在 PDF 1。6 中为线标记注释引入的维度意图类似。
- 为线，折线和多边形标注注释的维度指定单位和缩放的能力.该功能使用户可以测量文档中的距离,例如建筑图的宽度或 3D 横截面的直径。
- 在 3D 图形的特定视图上放置标记注释的功能。
- 锁定注释内容的功能。

在法律环境中使用的交互功能

标记注释的另外一个用于法律环境，特别是银行业务。添加指定打印特性的新的查看器首选项设置，如纸张选择和处理，页面范围，副本和缩放。当用户使用这些浏览器首选项设置打印 PDF 文档时，打印对话框将按照这些设置中的指定进行预先填充。这种能力增强了 PDF 文档的打印预览功能，这可以使 PDF 文档更适合在法律环境中使用。

1.2。3 可访问性相关功能

TaggedPDF 的增加标识了更多类型的页面内容的作用：

- 识别非交互式 PDF 文档中表单域的角色的能力。此更改标识按钮字段（按钮,复选框和单选按钮）和文本字段（填充或未填充)。
- 提供与表结构相关联的表摘要的能力。该功能可以帮助视障者了解表的目的和结构，而无需阅读该表中的内容.
- 识别背景页面工件的能力,这对文档回流可能很重要.背景工件是对作者的原始内容的意义无贡献的对象的集合,例如侧栏或全页背景图像后面的彩色矩形。这样的页面背景可能不与任何逻辑结构相关，但是它们可能在再现原始文档的外观方面是有用的.
- 区分分页工件的能力：水印、页眉和页脚。

1.2.4 文件导航功能

对文档导航的添加指定了可移植集合的查看和组织特性，其中多个文件附件显示在一个窗口内。可移植的集合用于显示、排序和搜索相关文档的集合，如电子邮件归档、照片收集和工程投标集。

1。2。5 安全相关功能

PDF1.7 引入的内容，增加了文档作者可以对数字签名和超出要求施加的控制。PDF 消费者应

用程序必须满足:

- 附加的数字签名约束，在签名应用时执行.这些约束包括优先杂凑算法，签名中使用的证书的撤销检查以及澄清其他参数解释的标志。
- 关于签署时使用的证书的其他限制.这些约束包括必须存在于证书中的主题可分辨名称（DN）字典，必须存在于签名证书中的 KeyUsage 扩展名以及澄清指定证书约束的其他参数的解释的标志。
- 在处理或显示 PDF 文档之前，可以指定要求处理程序来验证 PDF 消费者应用程序必须满足的某些要求.此功能提供了一种方法，确保与 PDF 文档的向后兼容性，PDF 文档可能包含 JavaScript 段以验证要求.在添加此功能之前,JavaScript 是执行此类需求检查的唯一方法。该功能确保 JavaScript 段验证要求,或者指定的处理程序验证要求。

1.2.6 一般功能

通过提供字符串和文件名的编码信息,PDF 1.7 的增加提供了更多的跨平台和跨应用程序稳定性:

- 描述用于字符串的编码的字符串类型的说明。在整个 PDF 引用中，字符串类型的任何用途都将替换为更具体的字符串类型之一。 此澄清不需要更改 PDF 消费者应用程序。相反，它更清楚地了解每个 PDF 字符串条目支持的编码. 当将 PDF 文档中的字符串与外部源（例如 XML 文档或 3D 图形）中的字符串进行比较时，这种理解可能尤为重要。
- 使用 Unicode 指定文件名的功能，以及使用正在查看文档的平台的标准编码指定文件名。此功能可以减少在不同平台或不同语言中编码的文件路径名称解码的问题。

1。2.7 PDF 规范更改

此版本的“PDF 规范”包含与新功能或附加功能无关的说明:

- 所有混合模式的公式说明。

- 嵌套目录条目或列表条目的 TaggedPDF 表示形式的说明。

1。3 相关文献

PDF 和 PostScript 页面描述语言共享相同的底层 Adobe 成像模型。可以在 PDF 和 PostScript 语言之间直接转换文档； 当打印时，两个表示产生相同的输出。然而，PostScript 包括一个不存在于 PDF 中的通用编程语言框架。PostScript 语言参考是 PostScript 语言及其成像模型的综合参考。

PDF 和 PostScript 程序支持的几种标准格式字体,包括 Adobe Type 1,CFF（Compact Font Format）,TrueType，OpenType 和 CIDkeyed 字体。这些字体的 PDF 表现在本书中有说明。然而，字体文件本身的规格是单独发布的，因为它们是高度专业化的,并且针对不同的用户群体会有不同.有各种 Adobe 文献说明了字体格式。参考文献列出了这些文献,以及与 PDF 相关的其他文献和本书的内容。

1.4 版权说明

Adobe 拥有 PDF 规范中的版权.Adobe 将强制执行其版权。Adobe PDF 必须保留其 PDF 版权的原因是为了保持便携式文件格式标准的完整性,并确保公众可以区分便携式文档格式和电子文档的其他交换格式.然而，Adobe 希望促进使用便携式文档格式来进行不同产品和应用程序之间的信息交换。因此，Adobe 以其版权的所有人授予复制，修改和分发书面规范中的任何示例代码的权限，以符合 PDF 规范的方式实施便携式文档格式所必需的程度。[1]

Adobe 系统公司及其子公司拥有涵盖 PDF 规范文献中披露的技术的多项专利。PDF 规范文献中的任何内容均不授予任何专利权.尽管如此，Adobe 希望鼓励在各种设备和平台上实施 PDF 计算机文件格式，因此为全球的 PDF 实施者提供了一些免版税专利许可。要查看这些许可证，请访问 http://www.adobe.com/go/developer_legalnotices.

1．该示例代码包括但不限于数字结构，操作符和 PostScript 语言功能定义的受版权保护的列表，在 PDF 参考，第五版，1。6 版，第 1.5 节（知识产权）中引用。

第 2 章 概述

2.1 图像模型

PDF 的核心是它描述复杂图形和排版的外观的能力.通过使用 Adobe 成像模型实现此功能,该模型与 PostScript 页面描述语言中使用的高级别，与设备无关的表示法相同。

虽然应用程序理论上可以将任何页面描述为全分辨率像素阵列，但是生成的文件对于高分辨率设备来说将是庞大的,取决于设备的，并且是不切实际的。高级成像模型使应用程序能够根据抽象图形元素而不是直接根据设备像素来描述包含文本、图形形状和采样图像的页面的外观。这样的描述是经济的和设备独立的,并且可以用于在广泛的打印机、显示器和其他输出设备上产

生高质量的输出。

2。1。1 页面描述语言

在其他角色中，PDF 使用页面描述语言，这是一种用于描述相对于成像模型的页面的图形外观的语言。应用程序通过两阶段过程产生输出:

1. 应用程序在页面描述语言中生成与设备无关的所需输出的描述。
2. 控制特定输出设备的程序解释该描述并将其呈现在该设备上。

这两个阶段可以在不同的地方和不同的时间执行。页面描述语言作为可打印或可显示文档的紧凑型设备独立传输和存储的交换标准。

2.1.2 Adobe 图像模型

Adobe 成像模型是从图形艺术中借鉴的二维图形的简单而统一的视图。 在此模型中，“画刷"放置在选定区域的页面上:

- 绘制的图形可以是字符形状(字形），几何形状，线条或采样图像的形式，例如照片的数字表示。
- 画刷可能是彩色或黑色,白色或任何阴影灰色.它也可能采取重复模式 (PDF 1.2)的形式或颜色之间的平滑过渡色(PDF 1。3)。
- 这些元素中的任何元素可以被剪切,使在它们被放置在页面上时，可以出现在其他形状内.

页面的内容流包含描述一系列图形对象的操作数和操作符。PDF 消费者应用程序维护一个隐含的当前页面,它累积绘画操作员创建的标记.最初，当前页面完全为空。对于在内容流中遇到的每个图形对象，应用程序将标记放置在当前页面上，该代码替换或组合它们可能覆盖的任何先前标记.一旦页面完全组合,则在输出介质上呈现累积的标记并且当前页面被清除为空白。

PDF 1。3 和早期版本使用一种不透明的成像模型,其中每一个新的图形对象被绘制在一个页面上，完全覆盖了这些位置上的原内容（受某些可选参数的影响，可能会修改这种行为;参见第

4.5。6 节“套印控制”.不管是什么颜色的对象——白色、黑色、灰色或彩色--都被放在页面上，就像用不透明的颜料涂在上面一样.PDF 1。4 引入了一个透明的图像模型，在该模型中，在页面上绘制的对象不需要完全不透明.相反，新绘制的对象与页面的先前存在的内容相结合,产生的结果将对象的颜色和其背景根据其各自的不透明度特征组合起来。透明成像模型在第 7 章中描述。

主要图形对象(包括其他)如下:

- 路径对象由一系列连接的和断开连接的点、线和曲线组成,它们一起描述形状和它们的位置.它是通过对路径构建操作符的顺序应用而构建的，每一个都附加一个或多个新元素。路径对象是由一个路径绘图操作符结束的，它以某种方式绘制页面上的路径.主要的路径绘制操作符是 S（描边）,它在路径上画一条线，f（填充)绘制路径的内部。
- 文本对象由一个或多个符号形状组成，代表文本的字符。字符的字形在一个叫做字体的单独的数据结构中被描述.像路径对象一样,文本对象可以被描边或填充。
- 图像对象是样本值的矩形数组，每个值代表矩形内特定位置的颜色。这些对象通常用来表示照片.

绘制图形需要各种参数，一些是显式的，另一些是隐式的。隐式参数包括当前颜色、当前行宽、当前字体(字体和大小）以及其他许多参数.这些隐式参数组合在一起构成了图形状态；在图形状态中,有用于设置每个隐式参数值的操作符。绘制图形时，调用当前生效的值。

图形状态中的一个额外的隐式参数会修改绘制图形对象的结果. 当前的剪切路径概述了可以放置画刷的当前页面的区域.虽然可以在当前页面的任何位置放置标记,但只有落在当前剪切路径内的标记才会影响页面； 那些落在外面的不会影响页面。最初,当前的剪切路径包含页面的整个可成像区域。它可以暂时被简化为由路径或文本对象定义的形状,或者多个这样的形状的交集。随后放置的标记被限制在该边界内。

2.1.3 光栅输出设备

Adobe 图像模型的大部分功能源自它处理一般光栅类输出设备的能力.这些技术包括激光、点阵、喷墨打印机、数字成像器和光栅扫描显示器等技术.光栅输出设备的定义特性是，一个打印或显示的图像由一个矩形阵列（或光栅)组成，这些点被称为像素(图像元素)，可以单独处理。在典型的 bilevel 输出设备中，每个像素都可以是黑白的。在某些设备上,像素可以设置为灰色或某种颜色的中间阴影。设置单个像素颜色的能力使生成打印或显示输出成为可能，这些输出可以包括文本、任意图形形状和采样图像。

光栅输出设备的分辨率可以测量沿两个线性尺寸的每单位距离的像素数。分辨率通常是水平和垂直，但不是一定的。制造商对设备技术和价格/性能的权衡，产生了特有的分辨率范围：

- 电脑显示器的分辨率相对较低，通常为每英寸 75 到 110 像素.
- 点阵式打印机通常每英寸有 100 到 250 个像素.
- 喷墨和激光扫描静电印刷技术实现了每英寸 300 至 1400 像素的中级分辨率。
- 摄影技术允许每英寸 2400 像素以上的高分辨率。

更高的分辨率可以产生更好的质量和保真度，但成本更高。随着技术的进步和计算成本的降低,产品逐渐演变为更高的分辨率。

2.1.4 扫描转换

通过称为扫描转换的过程,在光栅输出设备上渲染抽象图形元素（例如线，圆，字符字形或采样图像）。给定图形元素的数学描述，该过程确定要调整的像素以及为这些像素分配哪些值以在可用的设备分辨率下实现最忠实的再现.

页面上的像素可以由计算机内存中的像素值的二维数组表示。对于其像素只能为黑色或白色的输出设备，单个位足以表示每个像素.对于可以再现灰度级或彩色的设备,需要每像素多个位。

注意：虽然印刷或显示页面的最终表示在逻辑上是一个完整的像素阵列，但其在计算机存储

器中的实际表示不需要由每个像素的一个存储单元组成。一些实现使用其他表示，如显示列表。Adobe 成像模型经过精心设计,不依赖于光栅内存的任何特定表示.

对于要出现在页面上的每个图形元素，扫描转换器设置相应像素的值。当页面描述的解释完成时，存储器中的像素值表示页面的外观.此时，光栅输出过程可以在打印页面或显示屏幕上呈现此表示（使其可见）。

扫描转换图形形状（如矩形或圆形）需要确定哪些设备像素位于形状内，并适当地设置其值（例如,为黑色）。因为形状的边缘并不总是精确地落在像素之间的边界上，因此需要一些策略来决定如何沿边缘设置像素。扫描转换表示文本字符的字形在概念上与扫描转换任意图形的形状相同。然而，字符字形对可读性要求更加敏感，并且必须满足更严格的客观和主观质量测量。

在双层设备上渲染灰度元素通过称为半色调的技术来实现.根据某些图案将像素阵列分成小簇(称为半色调屏)。在每个集群中，一些像素被设置为黑色，而其他像素被设置为与页面上该位置所需的灰度级成比例的白色。当从足够的距离观察时，各个点变得不可察觉，感觉到的结果是灰色阴影.这使得二级光栅输出设备能够重现灰度，并近似诸如照片的自然图像。一些彩色设备使用类似的技术。

2.2 其他通用特性

本节介绍 PDF 的除了成像模型之外的其它需要注意的通用特性。

2。2。1 可移植性

PDF 文件被表示为 8 位二进制字节的序列.PDF 文件被设计成可移植到所有平台和操作系统。二进制表示的目的是直接生成、传输和使用,而不需要在本机字符集、行尾表示或其他平台上使用的其他约定之间进行转换.

任何 PDF 文件也可以以只使用 7 位 ASCII(美国信息交换标准编码)字符编码的形式表示。这对于展示的目的很有用，就像在这本书里一样。但是，这种表示并不推荐用于实际使用，因为

它比普通的二进制表示效率低。不管使用哪种表示,PDF 文件都必须作为二进制文件传输和存储,而不是作为文本文件。无意的更改，例如在文本结束的约定之间的转换，将损坏该文件，并可能使其无法使用。

2.2。2 压缩

为了减少文件大小，PDF 支持许多种行业标准压缩过滤器:

- JPEG 和（PDF 1。5)JPEG2000 彩色和灰度图像压缩.
- CCITT（组 3 或组 4），运行长度和(在 PDF 1.4）JBIG2 单色图像压缩。
- LZW(Lempel—Ziv—Welch）和（从 PDF 1.2 开始）Flate 压缩，文本，图形和图像的压缩。

使用 JPEG 压缩,颜色和灰度图像可以压缩 10 倍或更多.对单色图像的有效压缩取决于所使用的压缩过滤器和图像的属性,但降低 2:1 到 8:1 是常见的(或 20:1 至 50:1,这是 JBIG2 压缩的一页图片的图像).内容流的 LZW 或 Flate 压缩描述了文档中所有其他文本和图形的压缩比，约为 2:1。所有这些压缩过滤器都产生二进制数据，如果需要一个 7 位的 ASCII 表示，则可以进一步转换为 ASCII base - 85 编码。

2.2。3 字体管理

在文档交换中，管理字体是一个基本的挑战。通常，文档的接收者必须拥有与创建文档相同的字体。如果一个不同的字体被替换，它的字符集、字形形状和指标可能与原来的字体不同.这种替代可以产生意想不到的不需要的结果，比如文本的行延伸到边缘或与图形重叠。

PDF 提供了处理字体管理的各种方法:

- 原始的字体程序可以嵌入在 PDF 文件中,这确保了最可预测和可靠的结果。PDF 支持各种字体格式，包括 Type 1、TrueType、OpenType 和 cid — keyed 字体。

- 为了节省空间，可以嵌入一个字体子集,只包含那些在文档中实际使用的字符的符号描述。另外，Type 1 字体可以用一种特殊的紧凑格式表示。
- PDF 规定了一组 14 种标准字体,可以在没有预先定义的情况下使用.这些包括三个拉丁文字体(Courier，Helvetica *和 Times *）中的四个面,以及两个符号字体(Symbol 和 ITC ZapfDingbats®）。所有 PDF 消费应用程序都需要这些字体或具有相同指标的合适替代字体。
- PDF 文件可以通过名称引用未嵌入 PDF 文件的字体。在这种情况下，PDF 的使用者可以在其环境中使用这些字体.这种方法受到上面提到的不确定性的影响.
- PDF 文件包含其使用的每种字体的字体描述符。 字体描述符包括字体度量和样式信息,使得应用程序能够在必要时选择或合成合适的替代字体. 虽然字形的形状与预期的形状不同，但它们的位置是准确的。

字体管理主要关注生成文本的正确外观,即字形的形状和位置。 然而，PDF 应用程序有时需要提取文本的含义，以某些标准信息编码（如 Unicode）表示. 在某些情况下，可以从用于表示 PDF 文件中的文本的编码推导出此信息。否则,PDF 制作应用程序应通过包含特殊对象 ToUnicode CMap 来明确指定映射。

2。2.4 单次传递文件生成

由于系统限制和效率考虑,可能需要或希望应用程序在单次传递中生成 PDF 文件。例如，程序可能具有有限的可用内存或无法打开临时文件.因此，PDF 支持单次传递生成文件。虽然一些 PDF 对象必须以字节指定其长度,但是提供了一种机制，允许长度跟随 PDF 文件中的对象。此外，在生成所有页面之后,文档中的页数等信息可以写入文件。

在单次传递中生成的 PDF 文件通常不是为了最有效的阅读而排序的，特别是在通过网络访问文件的内容时。当生成要多次查看的 PDF 文件时,值得执行第二次传递来优化文件中发生的对象的顺序。PDF 指定了一个特定的文件组织，线性化 PDF,它在附录 F 中列出.其他优化也是可能的，

例如检测图形对象的重复序列，并将其折叠成只指定一次的共享序列。

2.2。5 随机访问

PDF 文件应该被认为是一个数据结构的扁平表示，它包含可以以任意方式相互引用的对象集合。在 PDF 文件中出现的对象顺序没有语义意义。一般来说，应用程序应该根据对象的引用来处理 PDF 文件，而不是按顺序处理对象.这对于交互式文档查看或任何在 PDF 文件中的页面或其他对象被从序列中访问的应用程序尤其重要。

为了支持对单个对象的这种随机访问，每个 PDF 文件都包含一个交叉引用表,可用于定位和直接访问文件中的页面和其他重要对象。交叉引用表存储在文件的末尾，允许在单次传递中生成 PDF 文件的应用程序可以轻松地存储它，而那些读取 PDF 文件的应用程序可以轻松地定位它。通过使用交叉引用表，找到一个页面或其他对象所需的时间几乎与文档的长度无关,从而可以有效地访问包含数百或数千页的 PDF 文档。

2。2。6 安全性

PDF 有两个可以在任何文档中单独或一起使用的安全功能：

- 该文档可以加密，以便只有授权的用户可以访问它.对文档的所有者和所有其他用户有单独的授权； 可以选择性地限制用户的访问以仅允许某些操作，例如查看，打印或编辑.
- 该文件可以进行数字签名,以证明其真实性。签名可以采取许多形式，包括用公共/私人密钥加密的文档摘要，诸如指纹的生物特征签名等.签名 PDF 文件的任何后续更改都会使签名无效。

2.2。7 增量更新

应用程序可能允许用户修改 PDF 文档。每次保存对文档的修改时,用户不必等待整个文件(可能包含数百页或更多页）。PDF 允许对文件进行修改,保留原始数据.增量更新文件时附加的附录

仅包含实际添加或修改的对象，并包含对交叉引用表的更新。增量更新允许应用程序以与修改大小成比例的时间量将文档的修改保存到 PDF 文档,而不是文件的大小。

另外，由于文件的原始内容仍然存在文件中,所以可以通过删除一个或多个附录来撤消保存的更改。当应用数字签名并随后需要验证时，恢复原始文档的确切内容的能力至关重要。

2。2。8 可扩展性

PDF 的设计是可扩展的。不仅可以添加新功能，而且基于早期版本的 PDF 的应用程序在遇到不了解的较新功能时可以合理地执行。附录 H 描述了 PDF 消费者应用程序在这种情况下的行为。

此外,PDF 提供了应用程序将其自己的私人信息存储在 PDF 文件中的方法。 当该文件由同一应用程序导入时,可以恢复该信息，但是被其他应用程序忽略。 因此，PDF 可以作为应用程序的本机文件格式,而其文档可以被其他应用程序查看和打印。特定于应用程序的数据可以作为标记内容存储在 PDF 内容流中注释图形对象，也可以作为与 PDF 内容未连接的完全独立的对象进行存储。

2。3 创建 PDF

PDF 文件可以直接由应用程序生成,也可以间接地通过其他文件格式或成像模型的转换来生成。随着 PDF 文档和处理它们的应用程序变得越来越普遍，将会产生新的创建和使用 PDF 的方法.

许多应用程序可以直接生成 PDF 文件，有些应用程序也可以导入它们。这种直接方法是优选的,因为它使应用程序能够访问 PDF 的全部功能,包括成像模型和交互式和文档交互功能.或者,直接生成 PDF 的应用程序可以间接产生 PDF 输出。有两种主要的间接方法:

- 应用程序通过调用应用程序编程接口(API）来描述其打印输出,例如 Microsoft®Windows® 的 GDI 或苹果 Mac OS 的 QuickDraw。一个称为打印机驱动程序的软件组件拦截这些调用，并解释它们以 PDF 格式生成输出。

- 应用程序可以直接以某种其他文件格式生成可打印的输出,例如 PostScript，PCL，HPGL 或 DVI，它们通过单独的翻译程序转换为 PDF。

虽然这些间接策略通常是从现有应用程序获取 PDF 输出的最简单方法，但是生成的 PDF 文件可能无法充分利用高级 Adobe 成像模型。这是因为应用程序的 API 调用或中间输出文件中体现的信息通常以太低的级别描述所需的结果。原始应用程序维护的任何更高级别的信息都已丢失，不适用于打印机驱动程序或转换器.

图 2.1 和 2.2 显示了 Acrobat 产品如何支持这些间接方法。在 Windows 和 Mac OS 平台上可用的 Adobe PDF 打印机(图 2。1）充当打印机驱动程序，通过操作系统的 API 拦截正在运行的应用程序生成的图形和文本操作.Adobe PDF 打印机将它们转换为等效的 PDF 操作符，并将它们嵌入到 PDF 文件中，而不是将这些操作转换为打印机命令并将其直接传输到打印机。结果是一个平台独立的文件，可以在任何支持的平台上运行的 PDF 查看器应用程序（如 Acrobat）进行查看和打印，即使是与最初生成文件的平台不同的平台。

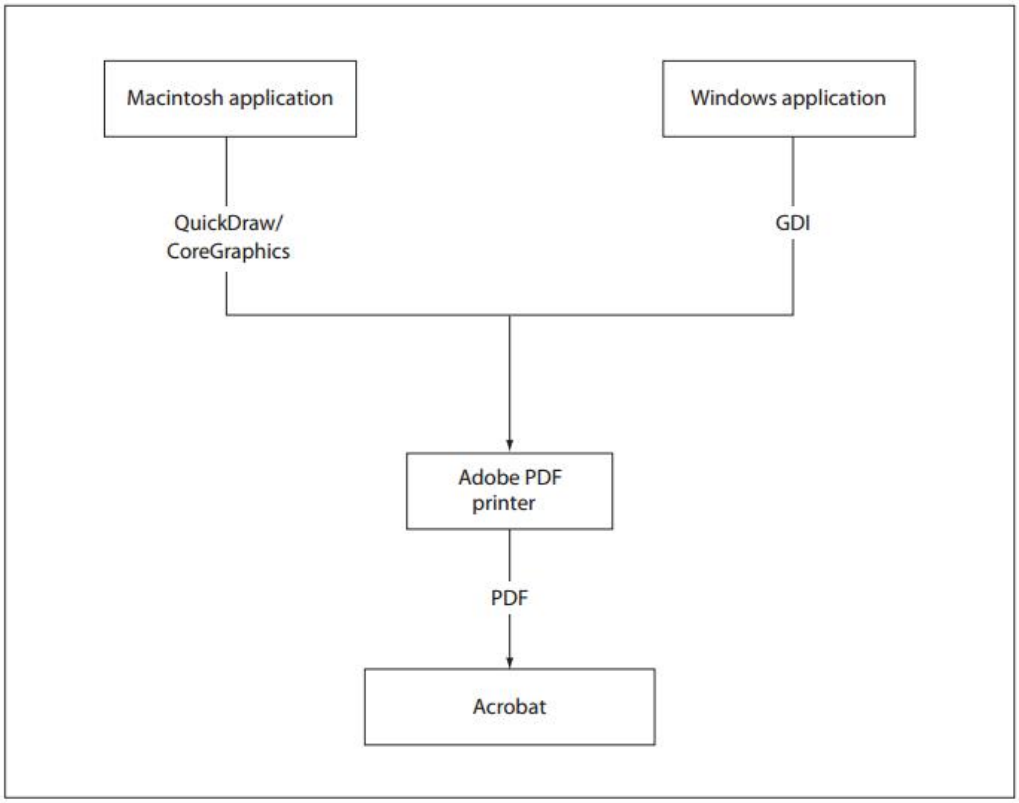

图 2.1 使用 Adobe 打印机创建 PDF 文件

而不是通过 API 调用来描述其可打印的输出，有些应用程序直接生成 PostScript 页面描述，或者是因为 QuickDraw 或 GDI 图像模型中的限制,或者因为应用程序运行在诸如 DOS 或 UNIX®等平台上，而没有系统级的打印机驱动程序存在。这些应用程序生成的 PostScript 文件可以使用 AcrobatDistiller®应用程序转换为 PDF 文件(参见图 2.2)。因为 PostScript 和 PDF 共享相同的 Adobe 成像模型，所以 Distiller 可以将 PostScript 文件的精确图形内容保留在 PDF 中。此外,Distiller 还支持 PostScript 语言扩展，称为 pdfmark，它允许生产应用程序在 PostScript 文件中嵌入指令，以创建超文本链接，逻辑结构以及 PDF 的其他交互和文档交互功能。同样，可以在任何支持的平台上使用查看器应用程序（如 Acrobat）查看生成的 PDF 文件。

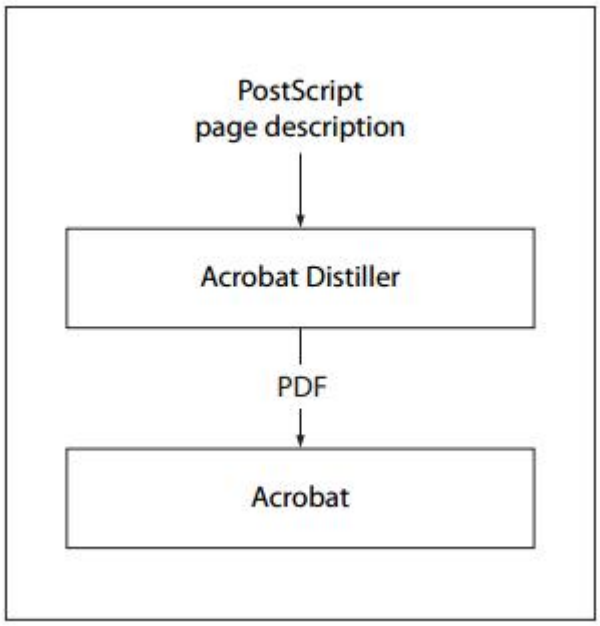

图 2。2 使用 Acrobat Distiller 创建 PDF 文件

2.4 PDF 和 PostScript 语言

用于设置图形状态和绘制图形对象的 PDF 操作符与 PostScript 语言中的相应操作符类似. 然而，与 PostScript 不同，PDF 不是一个全面的编程语言； 它降低了灵活性，提高了效率和可预测性。 因此 PDF 与 PostScript 的区别在于以下重要方面：

- PDF 强制执行严格定义的文件结构，允许应用程序以任意顺序访问文档的部分。
- 为了简化内容流的处理，PDF 不包括常见的编程语言功能,如过程，变量和控制结构。

- PDF 文件包含字体指标等信息，以确保查看保真度.
- PDF 文件可能包含与成像模型不直接相关的附加信息，例如用于交互式查看的超文本链接和用于文档交互的逻辑结构信息。

由于这些差异，PDF 文件通常不能直接传输到 PostScript 输出设备进行打印（尽管有些这样的设备也直接支持 PDF）。将 PDF 文档打印到 PostScript 设备的应用程序必须遵循以下步骤：

1. 插入包含 PostScript 过程定义的过程集以实现 PDF 操作符。
2. 提取每个页面的内容。每个内容流本质上是传统 PostScript 程序的脚本部分，使用非常具体的过程，例如 m 是 moveto 和 l 是 lineto。
3. 根据需要对压缩文本,图形和图像数据进行解码。PDF 中使用的压缩过滤器与 PostScript 中使用的压缩过滤器兼容； 它们可能或可能不受支持，具体取决于目标输出设备的语言级别。
4. 将任何所需的资源（如字体）插入到 PostScript 文件中.这些可以是基于 PDF 文件中的字体指标的原始字体或合适的替代字体。字体可能需要转换为 PostScript 解释器识别的格式，例如 Type 1 或 Type 42。
5. 以正确的顺序放置信息。结果是一个传统的 PostScript 程序，它完全代表文档的视觉方面，但不再包含 PDF 元素，如超文本链接，注释和书签。
6. 将 PostScript 程序传送到输出设备。

第 3 章 语法

本章介绍了 PDF 对象、文件和文档级别的语法，是后续章节的语法基础。后续章节介绍 PDF 文件的页面描述、交叉引用和文件逻辑结构。

PDF 语法分为 4 个部分，如图 3。1 所示：

- 对象。PDF 文档是一种数据集合,是由基本的数据对象组成。

3.1 节，“词汇术语(Lexical Conventions）”介绍用于写入对象和其他语法元素的字符集。

3。2 节，“对象(Objects）”介绍对象的语法和基本属性.

3。2。7 节,“流对象（Stream Objects）"介绍最复杂的数据类型流对象.

- 文件结构。文件结构定义了 PDF 文件中对象如何存储、访问和更新。文件结构与对象的语义无关.

  3。4 节,“文件结构(File Structure)”介绍文件结构。

  3。5 节，“加密(Encryption)”介绍文件级加密。

- 文档结构。PDF 文档结构描述了如何使用基本类型对象来表示 PDF 文档的组件，如页、字体、注释等等.

  3。6 节，“文档结构(Document Structure）"介绍文档结构，后面的章节介绍文档组件的详细语义。

- 内容流。PDF 内容流包含描述页面或其他图形实体外观的指令序列。这些指令，同时也被表示为对象，在概念上与表示文档结构的对象不同，并且分别描述。

  3.7 节，内容流和资源(Content Streams and Resources）介绍了内容流及其相关资源。

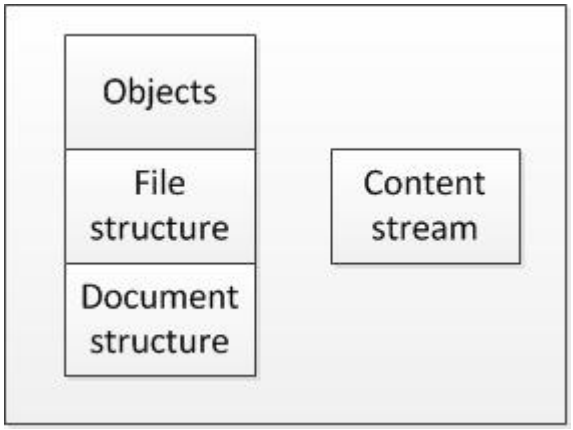

图 3。1　PDF 组件

此外，本章还介绍了一些由基本数据对象构成的数据结构，它们被广泛使用,因此基本上可以将它们当作基本数据对象。这些数据结构包括 3.8 节“通用数据结构”,3。9 节“函数”，3.10 节“文件规范"。

PDF 的对象和文件语法也被用作其他文件格式的基础.包括 8.6.6 节表单数据描述的表单数

据格式(FDF)和在 Adobe 的技术说明#5620 便携作业格式中描述的便携作业格式(PJTF)。

3.1 词汇约定

在最底层，PDF 文件是一个 8 位字节的序列。这些字节可以根据后面的语法规则分组为不同的标志（tokens）.一个或多个标志构成了更高级的语义实体，即对象。对象是构建 PDF 文档的基础数据值.

PDF 可以完全表示可见的 ASCII 字符，加上空白字符（如空格、制表符、回车及换行）。ASCII 是美国信息交互的标准编码格式，ASCII 编码是将特定的 128 个字符使用二进制编码。但是，PDF 文件不限于 ASCII 字符集,它可以包含任意的 8 位字节，但须遵守以下条件：

- 分隔对象和描述 PDF 文件结构的标志都是用 ASCII 字符集编码的，用作保留字和标准字典中的关键字名称。
- 某些类型对象(字符串和流）的数据值可以不需要完全用 ASCII 字符集。如果是说明或注释，最好使用 ASCII 字符集。但是，在实际应用中，有些数据本身就是二进制的，如图像,这样比较紧凑和高效.

包含二进制数据的 PDF 文件必需原原本本的保存和传输文件的所有字节，也就是说,作为一个二进制文件而不是一个文本文件。这样的文件不能移植到增加保留字符编码，最大行长度,行尾约定或其他限制的环境中。

注意：本章中，字符和字节是表示相同的含义，都是指 8 位字节。它是独立于任何上下文关系的逻辑意义。

3。1.1 字符集

PDF 字符集分为三类：常规字符、分隔符和空白字符.这个分类将字符串、流和注释外的字符组合分组为不同的标志。对于字符串、流和注释有其它不同的规则.

空白字符（见表 3。1）分隔了句子结构,如将名字、数字分隔开。除了在注释、字符串和流

的其他所有情况下，所有的空白字符都是等价的，PDF 会将多个连续的空白字符当作一个空白字符。

表 3.1 空白字符

十进制值	十六进制值	八进制值	名称
0	00	000	空(NUL)
9	09	011	Tab(HT)
10	0A	012	换行(LF)
12	0C	014	换页（FF)
13	0D	015	回车(CR)
32	20	040	空格（SP）

回车(CR)和换行(LF）符，也被称为换行字符，被视为行结束(EOF）标记。回车后紧跟换行符也作为行结束(EOF）标记.在大多数情况下，EOF 标记也作为空白字符。但是，有时行结束标记是必需的，因为后续的标志（tokens)必需在行首。

注意：本书中的例子在行中放置标志的用法是习惯用法，但是，例子中空白字符使用缩进仅是为了易读，在实际应用中，不建议这样做。

分隔符是一些特殊的字符，包含(，)，〈，>，［，］，{，}，/，%。它们分隔语法实体，如字符串、数组、名称和注释.这些字符中的任何一个都标志一个实体的结束，但字符本身不属于实体。

除了空白字符和分隔符之外的所有字符都称为常规字符。这些字符包括 ASCII 字符集外的 8 位二进制字符。连续的常规字符组成一个标志(token)。

注意:PDF 是区分大小写的，相应的大小写字母被认为是不同的。

3。1.2 注释

在字符串或流之外的任何地方出现的百分号符号 (%)，都被当作注释。注释由百分号和行结束标志之间的所有字符组成，包括常规字符、分隔符、空格和制表符。PDF 忽略注释，将其视为单个空白字符.也就是说,一个注释将其之前的标志和之后的标志分开。因此，下面的 PDF 片断

```
abc% comment ｛/%) blah blah blah
123
```

在语法上只相当于标志 abc 和 123。

注释(除了在 3.4 节“文件结构”中描述的%PDF-n.m 和%%EOF）没有任何语义。它们在编辑 PDF 文件的应用程序中不一定会被保存(见附录 H 中的注释 2）。特别是，没有 PostScript 文档结构约定(DSC)的 PDF 替代品.

3.2 对象

PDF 包含 8 种基本对象类型：

- 布尔值
- 整数和实数
- 字符串
- 名字
- 数组
- 字典
- 流
- 空对象

对象可能被标记，这样它们可以被其他对象引用。被标记的对象也称为间接对象。

以下部分描述各个对象类型,以及如何创建和引用间接对象.

3。2。1 布尔对象

PDF 提供由关键字 ture 和 false 标识的布尔对象。布尔对象可以用作数组元素和字典入口的值，也可以作为 PostScript 计算器函数的运算结果以及作为条件运算符(if 和 ifelse)中的操作数（见 3。9。4 节“类型 4(PostScript 计算器)函数”）。

3.2.2 数值对象

PDF 提供两种类型的数值对象:整数和实数。整数对象表示以 0 为中心的一段范围内的数学整数.实数对象近似于数学实数,但范围和精度有限.它们通常以定点形式而非浮点形式表示。数字的范围和精度受到 PDF 应用程序的计算机中使用的范围和精度限制。附录 C 给出了通常实施环境中的限制。

一个整数由一个或多个十进制数组成，可带一个符号。

123　43445　 +17-98　 0

上述值被解释为带符号的十进制整数，并被转化为整数对象。如果它超出了整数的范围,则被转化为实数对象。

一个实数一个或多个十进制数和一个可选符号和头,尾或小数点。

34.5　-3。62 +123。6　　4。-.002　0。0

上述值被解释为实数，并被转化为实数对象。如果超出了实数范围，则会报错。

注意:PDF 不支持含有非十进制格式（例如 16＃FFFE)或指数格式（例如 6.02E23）数字的 PostScript 语法。

在本书中，术语“数"是指一个对象，可以是整数对象或实数对象.当期望使用一个实数对象时,可能被一个整数对象替代了，整数对象会自动转化为一个等价的实数对象.例如，没有必要以实际格式使用实数 1.0,而整数 1 就足够了。

3.2.3 字符串对象

字符串对象由一系列字节（无符号整型，范围 0 到 255）组成，字符串对象不是整数对象，而是以更紧凑的格式存储。字符串的长度受到实施环境限制,参见附录 C。

字符串对象可由两种方式表示:

- 用括号 ()括起来的文本字符.参见下面的“文本字符串”.
- 用尖括号〈〉括起来的十六进制字符。参见下面的“十六进制字符串".

本节只描述将字符串写入字节序列的基本语法。字符串可以用于多种目的,可以有多种格式.当字符串用于某些特定目的(例如表示日期)时,就需要有对应的格式标准(参见 3。8。3 节“日期”）来实现这些目的。这样的格式只是用于解释字符串的格式约定，而不是对象类型。使用特定格式的字符串对象的定义描述了该格式的用法.

3。8。1 节“字符串类型”描述了用于字符串对象内容的编码方案。

文本字符串

文本字符串是包括在一对括号 ()内的任意数量字符。除了不对称括号和反斜杠字符外，任何字符都可以出现在字符串中。不成对括号和反斜杠字符需要特殊处理，字符串中成对的括号不需要特殊处理。

下面是合法的字符串:

```
(This is a string)
(Strings may contain new lines
and such.)
(Strings may contain balanced parentheses() and
special characters (*! &} ^% and so on).)
(The following is an empty string.)
```

()

（It has zero (0) length.）

在文本字符串中，反斜杠(\) 用作转义字符功能，如换行符、不可见的 ASCII 字符、不成对括号或反斜杠字符本身。反斜杠之后的字符决定其精确的含义（见表 3.2）。如果反斜杠字符后的字符不是表中的任何一个，则反斜杠会被忽略。

表 3.2 文本字符串中的转义字符

字符	**含义**
\n	换行(LF)
\r	回车(CR)
\t	水平制表符 (HT)
\b	回退 (BS)
\f	换页符 (FF)
\ (	左括号
\)	右括号
****	反斜杠
ddd	以八进制表示的字符 ddd

如果一个字符串太长,一行放不下，可以使用反斜杠符号将它分为多行。反斜杠和紧随其后的行结束标记不被视为字符串的一部分。例如:

（These \

two strings \

are the same.）

(These two strings are the same.)

如果文本字符串中含有行结束标记,并且没有使用反斜杠符号进行转义,那结果相当于\n(无

论行结束标记是回车还是换行，或者两者都有）。

(This string has an end-of—line at the end of it。

)

（So does this one.\n）

ddd 转义符提供了一种表示不可见 ASCII 字符的方法。例如:

(This string contains \245two octal characters\307。）

数 ddd 可由一个，两个或三个八进制数字组成，高阶溢出被忽略。如果字符串的下一个字符也是一个数字,则必需使用三个八进制数字，若不够三个，则前面加 0，例如:

(\0053)

该字符串包含两个字符，\005（Control-E），然后是一个数字 3。而下例中的两个字符串都只包含单个字符\053，一个加号字符(+).

(\053)

（\53)

这种表示法提供了一种使用 ASCII 字符指定 7 位 ASCII 字符集之外的字符的方法。但是，任何 8 位值都可能出现在字符串中。特别是当文档被加密（参见 3。5 节“加密”)时,它的所有字符串都被加密，并且通常包含任意的 8 位值。请注意，反斜杠字符仍然需要作为转义符来指定不成对括号或反斜杠字符本身。

十六进制字符串

字符串也可以以十六进制形式表示，这样在 PDF 文件中就可以包含任意二进制数据。十六进制字符串以十六进制数字(0—9 和 A—F 或 a-f)的序列表示,它们包含在一对尖括号内。(〈和〉):

〈4E6F762073686D6F7A206B6120706F702E>

每对十六进制数字表示字符串的一个字节。空白字符（如空格，制表符,回车符，换行符和换页符）被忽略.

如果十六进制字符串的最后数字丢失，也就是说，如果有奇数个数字,则最后一位被假定为 0.例如:

〈901FA3〉

是一个 3 字节的字符串,十六进制编码为 90，1F 和 A3.但是

〈901FA〉

是一个 3 字节的字符串，十六进制编码为 90,1F 和 A0。

3.2。4 名字对象

名字对象是由字符序列唯一定义的原子符号。这种特殊的定义，意味着由相同字符序列组成的任意两个名字对象，都是相同的对象.原子的概念意味着名字对象没有内部结构;虽然它是由字符序列定义的，但这些字符不被认为是该名字对象的元素。

斜杠(/）字符表示了一个名字对象。斜杠字符不是名字的一部分，而是前缀,表示后面的字符序列构成名字。斜杠字符与名字的第一个字符之间不能有空格字符.名字可能包含任意常规字符,但不包括分隔符和空白字符(参见 3.1 节“词汇约定”）。大写和小写的字母被认为是不同的：/A 和/a 是不同的名字.以下的示例是有效的文本名字:

/Name1

/ASomewhatLongerName

/A;Name_With-Various＊＊*Characters?

/1.2

/＄$

/@pattern

/.notdef

注意:标志/（后面没有任何常规字符)也是一个有效的名字。

从 PDF 1。2 开始,除空字符(字符代码 0）以外的任何字符，都可以通过在它的两位十六进制编码前面加上数字符号（#），让其包含在一个名字中；见附录 H 的执行说明 3 和 4。此语法可以表示任何分隔符或空格字符或数字符号本身，对于编码不在范围 33(!)~126 (~)的字符，建议使用此语法表示。表 3。3 中的示例是 PDF 1。2 及更高版本中的有效文本名字。

表 3.3 使用#字符的文本名字示例

文本名字	结果
/Adobe#20Green	Adobe Green
/PANTONE＃205757＃20CV	PANTONE 5757 CV
/paired＃28＃29parentheses	paired()parentheses
/The_Key_of_F＃23_Minor	The_Key_of_F＃_Minor
/A#42	AB

名字的长度受实施环境的限制，见附录 C.该限制是针对名字内部的实际字符个数。例如,名字/ A＃20B 是四个字符（/,A，空格，B），而不是六个。

如上所述，名字对象在 PDF 文件中被视为原子符号。通常,构成名字的字符序列从不会被当作文本显示给人看或输出给外部程序。然而，偶尔需要将名字对象视为文本,例如表示一个字体名称(请参见表 5。8 中的 BaseFont 条目）或结构类型（请参见 10。6.2 节“结构类型"）.

在这种情况下，建议采用 UTF—8 编码来表示名字的字符序列。UTF—8 编码是一种 Unicode 的可变长度字节编码。ASCII 码中的可见字符的编码与 UTF-8 相同.这使得名字对象能够以任何自然语言表示文本，但需要对名字的长度实施限制。 （见附录 H 中的实施说明 5）。

注意：PDF 不规定选择哪一种 UTF—8 序列来将任意指定的外部文本表示为名字对象。在某些情况下，多个 UTF-8 序列可以表示相同的逻辑文本。由不同字节序列定义的名字对象在 PDF 中构成不同的名字对象，尽管 UTF—8 序列可能具有相同的外部解释。

在 PDF 中，名字对象始终以斜杠字符（/）开始，与诸如 true,false 和 obj 的关键字不同。

本书遵循一种排版约定,当它们出现在动态的文本和表格中时，无需使用前缀斜杠字符即可输入名字.例如，以 Type 和 FullScreen 表示的名字，在 PDF 文件（和本书中的代码示例）中以/ Type 和/ FullScreen 名字对象表示。

3。2。5 数组对象

数组对象是按顺序排列的对象的一维集合。与许多其他计算机语言的数组不同,PDF 数组可能是异构的； 也就是说，数组的元素可以是数字，字符串，字典或任何其他对象（包括其他数组）的任何组合。数组中的元素数量受制于实施限制；见附录 C。

数组是一个包含在一对方括号（ [和]）内的对象序列：

[549 3。14 false (Ralph) /SomeName]

PDF 直接支持一维数组。可以通过使用数组作为数组的元素来构建更高维度的数组,可嵌套到任何深度.

3。2。6 字典对象

字典对象是包含成对对象（称为字典条目）的关联表。每个条目的第一个元素是键，第二个元素是值。键必须是一个名字(与 PostScript 中的字典键不同，它可能是任何类型的对象）。值可以是任何类型的对象，包括另一个字典。值为空的字典条目(参见 3。2.8 节“空对象”）等同于缺少条目。(这与 PostScript 不同，其中空的行为与任何其他对象一样，作为字典条目的值）.字典中的条目数量受制于实施限制；见附录 C.

注意：一个字典中不应该有两个条目具有相同的键。如果一个键出现多次，它的值是未定义的。

字典对象是包含在一对双尖括号（〈< 〉>）内的一系列键-值对序列。例如：

```
<< /Type /Example
   /Subtype /DictionaryExample
   /Version 0。01
```

```
/IntegerItem 12
/StringItem （a string）
/Subdictionary 〈〈 /Item1 0。4
                /Item2 true
                /LastItem （not!）
                /VeryLastItem （OK ）
               >>
>〉
```

注意：请勿将双尖括号与单尖括号（<和〉）混淆，单尖括号是定义十六进制字符串的(请参见 3.23 节中的“十六进制字符串”）.

字典对象是 PDF 文档的主要构建块。它们通常用于集成并将复杂对象（例如文档的字体或页面）的属性与字典中的每个条目相关联,以指定属性的名称和值。按照惯例，这样的字典的类型条目标识字典描述的对象的类型。在某些情况下，使用子类型条目（有时缩写为 S)进一步标识一般类型的专门子类型。类型或子类型条目的值始终是一个名字。例如，在字体字典中，类型条目的值始终为字体，而子类型条目的值可以是类型 1，全真类型（TrueType)或其他几个值之一。

类型条目的值几乎可以总是从上下文推断出来。例如，Tf 操作符的操作数必须是字体对象；因此，字体字典中的类型条目主要用作记录和错误检查的信息.类型条目不是必需的，除非在其描述中另有说明； 但是，如果条目存在，它必须具有正确的值。此外，任何字典中的类型条目的值，即使是私有数据，都必须是本书中定义的名字或已注册的名字；详见附录 E.

3。2。7 流对象

流对象，和字符串对象一样,是一个字节序列。但是，PDF 应用程序可以逐步读取流，而字符串必须全部读取.此外，流可以是无限长度的,而字符串受到实施限制的。因此，具有潜在大量

数据的对象(如图像和页面描述）被表示为流.

注意：与字符串一样，本节只描述将流的字节序列语法。这些字节表示的含义由引用流的上下文决定。

一个流对象,由字典和包含在 stream 和 endstream 关键字之间的 0 个或多个字节组成.例如:

dictionary

stream

… Zero or more bytes…

endstream

所有流对象必须是间接对象(请参见第 3。2.9 节“间接对象"），而流对象中的字典对象(流字典）必须是直接对象。跟在字典后的关键字 stream，后面必需跟行结束标记(EOF),行结束标记可以是回车加换行或单独的换行,但不能是单独的回车。组成流对象的字节序列位于 stream 和 endstream 关键字之间； 流字典指定字节流的确切字节数。建议在流数据和 endstream 关键字之间要有行结束标记（EOF)，此标记不包括在流数据中。

另外，从 PDF 1。2 开始，字节可以包含在外部文件中，在这种情况下,流字典指定文件，并且忽略 stream 和 endstream 之间的任何字节。（见附录 H 中的说明 6）.

注意:stream 关键字后的行结束标记不能使用单独的回车，是因为，如果单独的回车可作为行结束标记，那就不能区分一个使用回车换行作为行结束标记的流和一个以回车作为行结束标记但以换行符为第一个数据字节的流。

表 3。4 列出了所有流字典中常见的条目；某些类型的流可能有额外的字典条目，比如说明流位置的条目。关于流对象的可选条目过滤器，指明了流数据在使用前是否需要转换或解码，以及如何转换或解码。过滤器在 3。3 节“过滤器”中进一步描述。

流长度

每个流字典都有一个长度（Length）条目，表示用于流数据的 PDF 文件字节数。(如果流有

过滤器,则长度是编码后数据的字节数。）此外,大多数过滤器被定义为使数据具有自限性；也就是说，采用一种编码方案,使用显式的数据终止（EOD）标记界定数据的范围。最后，流用于表示许多对象，可以从属性中推断长度。所有这些约束必须是一致的。

例如，使用单颜色分量和每个分量 8 位的 10 行 20 列的图像需要 200 字节的图像数据。如果流使用过滤器，PDF 文件中必需要有足够的生成这 200 字节数据的编码数据字节.如果长度太小或数据终止标记（EOD)标记太早出现或解码后数据不够 200 字节，则会发生错误。

如果流包含太多的数据，也是错误的，除了 PDF 文件中有额外的行结束标记在关键字 endstream 之前。

表 3.4 流字典常见条目

键	类型	值
Length	整型	（必需）从关键字 stream 之后的行开始到关键字 endstream 之前的最后一个字节的字节数.（可能还有一个额外的行结束(EOL）标记,不包括在计数中，逻辑上不属于流数据）.参见上面的“流长度”。
Filter	名字或数组	（可选）过滤器的名字或名字数组.过滤器用于处理关键字 stream 和 endstream 之间的流数据。多个过滤器应按他们的使用顺序列出。
DecodeParms	字典或数组	（可选）参数字典或此类字典的数组。参数字典是键Filter 中的过滤器使用.如果只有一个过滤器和该过滤器有参数，则 DecodeParms 必须设置为过滤器的参数字典,除非所有过滤器的参数都有默认值，这种情况下可忽略 DecodeParms 条目.如果有多个过滤器,并且有

		任意一个过滤器的参数设置为非默认值，则 DecodeParms 必需是每个过滤器条目的数组：条目是该过滤器的参数字典或空对象（该过滤器没有参数或所有参数都有默认值)。如果没有一个过滤器有参数，或者其所有参数都有默认值，则可以省略 DecodeParms 条目.（见附录 H 中的说明 7）
F	文件描述	（可选，PDF1.2）包含流数据的文件。如果此条目存在,则忽略关键字 stream 和 endstream 之间的字节.过滤器由 FFilter 指定而不是 Filter,过滤器参数由 FDecodeParms 指定而不是 DecodeParms.但是，Length 条目仍然指定 stream 和 enstream 之间的字节数量.(通常，stream 和 endstream 之间没有字节，长度为 0)（见附录 H 中的说明 46）.
FFilter	名字或数组	（可选，PDF1.2）过滤器的名字或名字数组。用于处理流数据定义在外部文件的情况。规则与 Filter 相同。
FDecodeParms	字典或数组	（可选,PDF1。2）参数字典或此类字典的数组。用于过滤器由 FFilter 定义的情况。规则与 DecodeParms 相同.
DL	整型	（可选； PDF 1。5）非负整数，表示解码后流中的字节数。例如,它可以用于确定是否有足够的磁盘空间可用于将流写入文件.这个值

只是一个提示；对于一些流过滤器，可能无法精确地确定此值。

3。2.8 空对象

空对象的类型和值与其他任何对象都是不同的.空对象的类型为空，由关键字 null 表示.间接对象（见 3.2.9 节“间接对象”）若引用了一个不存在的对象,则此间接对象与空对象相同。将空对象指定为字典条目的值(3。2。6 节“字典对象”)相当于完全省略该条目。

3.2。9 间接对象

PDF 文件中的任何对象都可能被标记为间接对象。给对象赋予一个唯一的对象标识符，其他对象就可以引用它(例如，作为数组的元素或字典条目的值）。 对象标识符由两部分组成：

- 对象编号，为正整数。间接对象通常在 PDF 文件中顺序编号,但不是必需的； 对象编号可以任意顺序分配。
- 生成编号，为非负整数.在新创建的文件中，所有的间接对象的生成编号都为 0.当文件以后更新时可能会引入非零生成编号； 请参见 3。4。3 节“交叉引用表"和 3。4.5 节“增量更新”.

所以，对象编号和生成编号的组合唯一的标识了间接对象。对象在其存在期间保留相同的对象编号和生成编号，即使其值被修改。

PDF 文件中间接对象的定义,包含对象编号、生成编号以及关键字 obj 和 endobj 之间的对象值。例如，下面

```
12 0 obj
   （Brillig）
endobj
```

定义了一个间接流对象，对象编号为 12，生成编号为 0，对象值为 Brilling.

可以通过由对象编号,生成编号和关键字R组成的间接引用在文件中的其他位置引用该对象:

```
12 0 R
```

从PDF1。5开始，间接对象可以放在对象流中（见3。4.6节“对象流”）。引用它们的方式完全相同,但是，在对象流中不使用关键字obj和endobj。

对未定义对象的间接引用不是错误的，它被简单的认为是对空对象的引用。例如，如果文件中有间接引用17 0 R，但没有对应的定义

```
17 0 obj
    …
endobj
```

那么间接引用被认为是对空对象的引用。

注意：在构成PDF文档的数据结构中，需要将某些值指定为间接对象引用。除了显式调用之外，任何对象（除了流）都可以直接指定或作为间接对象引用来指定；语义完全等价。特别要注意的是，定义文档可视内容的内容流不能包含间接引用。（见3.7。1节“内容流”）。另见附录H中的说明8.

示例3.1显示了使用间接对象来指定流的长度.在文件中，流的长度条目是跟在流后面的整数对象。这允许在一次传递中生成PDF格式的应用程序延迟指定流的长度，直到其内容生成为止.

示例 3.1

```
7 0 obj
    << /Length 8 0 R 〉〉                % 对对象8的间接引用
stream
    BT
        /F1 12 Tf
        72 712 Td
```

```
        (A stream with an indirect length) Tj
    ET
endstream
endobj
8 0 obj
    77                                  % 前面流的长度
endobj
```

3。3 过滤器

流过滤器在 3。2.7 节“流对象”中介绍.过滤器是流规范中的可选部分，指示流中的数据在使用之前必须进行解码.例如，如果流具有 ASCIIHexDecode 过滤器,则读取该流中数据的应用程序，需要将流中的 ASCII 十六进制编码数据转换为二进制数据。

生成 PDF 文件的应用程序可以对某些信息（例如，采样图像的数据）进行编码，将其压缩或转换为可移植的 ASCII 编码。然后，读取 PDF 文件的应用程序可以调用相应的解码器，将信息转换回其原始格式.

过滤器或流过滤器由流字典中的 Filter 条目指定(如果流是外部的话，则由 FFilter 条目指定)。过滤器可以级联形成一个管道,它通过序列中的两个或多个解码转换传递流。例如，可以使用以下流字典中的条目对使用 LZW 编码和 ASCII base—85 编码（按顺序)编码的数据进行解码:

```
/Filter  [/ASCII85Decode /LZWDecode]
```

有些过滤器可能需要参数来控制它们的运行方式。这些可选参数由流字典中的 DecodeParms 条目指定（如果流是外部的话,则由 FDecodeParms 条目指定)。

PDF 支持的标准过滤器，分为两大类:

- ASCII 过滤器可以对 ASCII 文本编码的任意 8 位二进制数据进行解码。(见 3。1 节“词汇约

定", 以了解此类编码用途。）请注意，ASCII 过滤器在加密的 PDF 文件中没有任何用途；参见 3。5 节“加密”。

- 解压过滤器可对压缩过的数据进行解码.压缩数据始终为 8 位二进制格式，即使原始数据为 ASCII 文本.（压缩对于大型采样图像特别有价值，因为它减少了存储空间要求和传输时间。某些类型的压缩是有损的，这意味着一些数据在压缩过程中会丢失，导致数据解压后损失精度。没有数据丢失的压缩称为无损压缩。）

标准过滤器总结在表 3。5 中,表中也指明过滤器是否接受任何可选参数。接下来的章节将详细介绍这些过滤器及其参数（如果有的话),包括一些过滤器的编码算法规范。(另见附录 H 中的说明 9 和 10）.

示例 3。2 显示了一个包含页标记指令的流,它使用 LZW 压缩算法压缩,然后以 ASCII base-85 编码表示。示例 3.3 显示了相同流，但没有任何编码。（该流的内容在 3。7.1 节“内容流"中进行了说明,第 5 章进一步描述了使用的操作符。）

表 3.5 标准过滤器

过滤器名称	**参数**	**描述**
ASCIIHexDecode	无	对以 ASCII 十六进制格式编码的数据进行解码，再现原始二进制数据。
ASCII85Decode	无	对以 ASCII base—85 十六进制格式编码的数据进行解码,再现原始二进制数据.
LZWDecode	有	解压使用 LZW（Lempel—Ziv-Welch)自适应压缩算法编码的数据，再现原始文本或二进制数据。
FlateDecode	有	（PDF 1.2）解压使用 zlib / deflate 压缩方法编码的数据，再现原始文本或二进制数

		据。
RunLengthDecode	无	解压使用面向字节的行程长度压缩算法编码的数据，再现原始文本或二进制数据（通常为单色图像数据，或任何包含频繁长时间运行单字节值的数据）。
CCITTFaxDecode	有	解压使用 CCITT 传真标准编码的数据，再现原始数据（通常为每像素 1 位的单色图像数据）。
JBIG2Decode	有	（PDF 1.4）解压使用 JBIG2 标准编码的数据,再现原始单色(每像素 1 位)图像数据(或该数据的近似值)。
DCTDecode	有	解压使用基于 JPEG 标准的 DCT(离散余弦变换）技术压缩的数据,再现近似原始数据的图像采样数据。
JPXDecode	无	(PDF 1。5)解压使用基于小波变换的 JPEG2000 标准压缩数据，再现原始图像数据。
Crypt	有	（PDF 1。5)解密由安全处理程序加密的数据，再现加密前的原始数据。

示例 3。2

```
1 0 obj
  〈〈 /Length 534
   /Filter  [/ASCII85Decode /LZWDecode]
  〉〉
```

stream

J..) 6T`?p&<!J9%_ [umg" B7/Z7KNXbN' S+,*Q/&"OLT' F

LIDK#!n`$"<Atdi`\Vn%b%)&'cA*VnK\CJY (sF〉c! Jnl@

RM] WM;jjH6Gnc75idkL5]+cPZKEBPWdR>FF (kj1_R%W_d

&/jS! ; iuad7h? [L-F$+]] OA3Ck*$IOKZ? ;〈) CJtqi65Xb

Vc3\n5ua:Q/=0$W〈#N3U;H, MQKqfg1? :lUpR;6oN [C2E4

ZNr8Udn。'p+?#X+1〉OKuk$bCDF/ (3fL5] Oq)^kJZ!C2H1

'TO] Rl? Q:&'〈5&iP!$Rq;BXRecDN [IJB`,)o8XJOSJ9sD

S] hQ; Rj@!ND)bD_q&C\g: inYC%)&u#: u,M6Bm%lY!Kb1+

": aAa'S`ViJglLb8<W9k6Yl\\OMcJQkDeLWdPN? 9A'jX*

al>iG1p&i;eVoK&juJHs9%;Xomop" 5KatWRT" JQ#qYuL,

JD? M$OQP) lKn06l1apKDC@\qJ4B! !(5m+j.7F790m(Vj8

8l8Q:_CZ(Gm1%X\N1&u! FKHMB~〉

endstream

endobj

示例 3。3

1 0 obj

〈〈 /Length 568 〉〉

stream

2 J

BT

/F1 12 Tf

0 Tc

0 Tw

```
72.5 712 TD
[(Unencoded streams can be read easily) 65 (, ) ] TJ
0 -14 TD
[(b) 20 (ut generally tak) 10 (e more space than \311)] TJ
T* (encoded streams.) Tj
0 -28 TD
[(Se) 25 (v) 15 (eral encoding methods are a) 20 (v) 25 (ailable in PDF) 80 (.)]
TJ
0 -14 TD
(Some are used for compression and others simply) Tj
T* [ (to represent binary data in an ) 55 (ASCII format.) ] TJ
T* (Some of the compression encoding methods are \
suitable ) Tj
T* (for both data and images, while others are \
suitable only ) Tj
T* (for continuous-tone images。) Tj
ET
endstream
endobj
```

3.3。1 ASCIIHexDecode 过滤器

ASCIIHexDecode 过滤器对以 ASCII 十六进制格式编码的数据进行解码。ASCII 十六进制编码和 ASCII base-85 编码(下一节描述)将二进制数据（如图像数据）转换为 7 位 ASCII 字符。一般来说，ASCII base—85 编码优于 ASCII 十六进制编码，因为它更紧凑：它的数据扩展倍数为

4:5,而 ASCII 十六进制编码为 1:2。

ASCIIHexDecode 过滤器为每对 ASCII 十六进制数字(0—9 和 A—F 或 a—f）生成一个字节的二进制数据。忽略所有的空格字符（见 3。1 节“词汇约定”）。以右尖括号(〉）为数据结束符（EOD）。任何其他字符都会导致错误。如果过渡器在读取奇数个十六进制数字后遇到 EOD 标记,则在最后一个数字后补 0 处理.

3.3。2 ASCII85Decode 过滤器

ASCII85Decode 过滤器对以 ASCII base-85 编码的数据进行解码,并生成二进制数据。下面描述了 ASCII base-85 中二进制数据的编码过程； ASCII85Decode 过滤器反转此过程.

ASCII base—85 编码使用字符！和字符 u 之间的字符以及字符 z,并以 2 个顺序字符～>作为其 EOD 标记。ASCII85Decode 过滤器忽略所有空格字符（见 3.1 节“词汇约定”）。任何其他表示 ASCII base-85 编码中不可能的组合的字符和字符序列都会导致错误。

具体来说，ASCII base—85 编码为每 4 字节的二进制数据产生 5 个 ASCII 字符。每组 4 个二进制输入字节 (b_1 b_2 b_3 b_4)被转换成一组 5 个输出字节(c_1 c_2 c_3 c_4 c_5),即 4 字节组数据转换成 5 字节组数据（数据扩展倍数 4：5)。使用关系

$$(b_1 \times 256^3) + (b_2 \times 256^2) + (b_3 \times 256^1) + b_4 =$$

$$(c_1 \times 85^4) + (c_2 \times 85^3) + (c_3 \times 85^2) + (c_4 \times 85^1) + c_5$$

换句话说，将 4 字节的二进制数据从 base—256 的数，转换为 base—85 的数。然后将 base-85 数的五个字节转换为 ASCII 字符,每个字符加上 33（字符!的 ASCII 码）.所生成的编码数据仅包含可见的 ASCII 字符，其 ASCII 码范围为 33（!）至 117（u）。一种特殊情况，如果所有五个字节都为 0，则它们由字符 z（ASCII 码 122)而不是五个感叹号(!! !!! ）表示.

如果要编码的二进制数据的长度不是 4 字节的倍数,则使用最后的局部 4 字节组数据来产生最后的局部 5 字节组数据。给出 n(1,2 或 3）字节的二进制数据，编码器首先添加 4 - n 个零字节，使其成为一个完整的 4 字节组数据，然后正常对该组进行编码得到完整的 5 字节组数据,但

特殊情况字符 z 不适用。最后，它只取所得到的 5 字节组数据中的前 n+1 个字符。这些字符后紧跟着 ~> EOD 标记.

以下情况（在正确编码的字节序列中不会出现）在解码期间会导致错误：

- 5 字节组数据表示的值大于 2^{32} - 1.
- z 字符出现在一个 5 字节组的中间。
- 最后的 5 字节局部组只包含一个字符.

3。3。3 LZWDecode 和 FlateDecode 过滤器

LZWDecode 过滤器和(PDF1.2) FlateDecode 过滤器有很多共同点，所以在本节中一并讨论。它们分别对使用 LZW 压缩算法和 Flate 压缩算法压缩的数据进行解压：

- LZW(Lempel-Ziv-Welch）是一种可变长度的自适应压缩算法，已被采用为标签图像文件格式（TIFF)标准中的标准压缩算法之一。LZW 压缩算法细节后面描述。
- Flate 算法是基于公共领域的 zlib/deflate 压缩算法,这是一种可变长度的 Lempel—Ziv 自适应压缩算法，与自适应霍夫曼编码算法一起级联。它在互联网 RFC 1950,ZLIB 压缩数据格式规范和 1951 年，DEFLATE 压缩数据格式规范（参见参考书目)中完整定义.

这两种压缩算法都可以压缩二进制数据或 ASCII 文本，并总是生成二进制数据（如所有压缩算法），即使原始数据是文本.

LZW 和 Flate 压缩算法可以检测和应用原数据中的多种格式,无论数据是文本还是图像。如后所述，两个过滤器都通过预测函数来支持数据的可选变换，这改进了采样图像数据的压缩。由于其级联的自适应霍夫曼编码，Flate 压缩的结果通常比相同数据使用 LZW 压缩的结果更紧凑.Flate 和 LZW 解压速度相当，但 Flate 压缩比 LZW 压缩慢得多。

通常，Flate 和 LZW 算法都大大压缩了原始数据.然而，在最坏的情况下(其中没有一对相邻的字符出现两次），Flate 编码扩展其原始数据不超过 11 个字节或 1。003 倍(以较大者为准），加上由 PNG 预测变量添加的算法标签的影响。对于 LZW 编码，最好的情况(全零）为大文件提供

了接近 1365:1 的压缩比，但是最坏情况的扩展至少是 1.125 倍，在一些实现中可能增加到近 1。5,加上 PNG 标签与 Flate 编码一样。

LZW 算法详述

使用 LZW 压缩算法编码的数据由 9 到 12 位长的编码序列组成. 每个编码表示输入数据（0-255），清除表标记(256），EOD 标记（257）的单个字符或表示先前在输入中遇到的多个字符序列的表项(258 以上）。

最初,编码长度为 9 位,LZW 表仅包含 258 个固定编码的条目。 随着编码的进行，条目被附加到表中，将新的编码与更长和更长的输入字符序列相关联. 编码器和解码器保持该表的相同副本.

当编码器和解码器各自（但同步）都意识到当前码长度不足以表示表中的条目数，它们将每个码的位数增加 1.在创建表条目 511 之后的第一个输出码是 10 比特位长，并且类似地对于 11（1023）和 12（2047）位。 代码永远不会超过 12 位；因此，条目 4095 是 LZW 表的最后一个条目.

编码器执行以下步骤序列以生成每个输出代码：

1. 累加与表中已存在的序列匹配的一个或多个输入字符的序列。为了最大化压缩，编码器寻找最长的这样的序列。
2. 发出与该序列相对应的代码。
3. 为第一个未使用的代码创建一个新的表条目.其值是在步骤 1 中找到的序列，后跟下一个输入字符.

例如,假设输入包含以下 ASCII 字符代码序列：

45 45 45 45 45 65 45 45 45 66

从空表开始，编码器按照表 3.6 所示进行.

表 3。6 典型的 LZW 编码序列

输入序列	**输出编码**	**加入表的编码**	**新编码序列号**
—	256（清除表）	-	-
45	45	258	45 45
45 45	258	259	45 45 45
45 45	258	260	45 45 65
65	65	261	65 65
45 45 45	259	262	45 45 45 66
66	66	—	-
-	257（EOD）	—	-

编码被打包成连续的位流，高位优先。然后将该流分为 8 位字节，高位优先。因此，编码可以任意地跨越字节边界。在 EOD 标记（代码值 257）之后，最后字节中的剩余位被设置为 0。

在上面的例子中,所有的输出代码都是 9 位长；它们将按如下(以十六进制表示)包装成字节：

80 0B 60 50 22 0C 0C 85 01

为了适应变长的输入序列，编码器可以在任何时候发出可清除的代码，这导致编码器和解码器使用初始表和 9 位代码长度重新开始.通常，编码器通过发出清除表码开始。当表格变满时，它必须发出一个清除表的代码； 也可能会更早发出。

LZWDecode 过滤器和 FlateDecode 过滤器的参数

LZWDecode 和 FlateDecode 过滤器接受可选参数来控制解码过程。这些参数中的大多数与减少压缩采样图像（颜色值的矩形阵列，如第 4。8 节“图像"中所述）大小的技术相关。例如,图像数据通常从样本到样本的变化很小.因此，去除相邻样本的值（称为差分的处理），并且对差异值进行编码而不是原始采样值，可以减小输出数据的大小.此外，当图像数据包含每个样本的几个颜色分量（红 — 绿 - 蓝或青色 - 黄 - 黑)时，取相邻样本中的相应分量值之间的差，

而不是在相同样本中的不同颜色分量之间,减少输出数据大小。

表 3.7 显示了 LZWDecode 和 FlateDecode 过滤器指定的可选参数.除非另有说明，否则任何可选参数提供给解码过滤器的值必须与数据编码时所使用的值相匹配.

表 3。7 LZWDecode 和 FlateDecode 过滤器的可选参数

键值	类型	说明
Predictor	整型	选择预测算法的代码，如果有的话.如果该条目的值为 1，则过滤器假设使用普通算法对数据进行编码,而无需预测。如果值大于 1，则过滤器假定在编码之前数据被差分，并且 Predictor 选择预测器算法。 有关 Predictor 值大于 1 的更多信息,请参阅下面的“LZW 和 Flate Predictor 函数”。默认值：1.
Colors	整型	（仅当 Predictor 大于 1 时使用）每个样本的交错颜色分量数.PDF 1。2 或更早版本中的有效值为 1 到 4,PDF 1。3 或更高版本中的有效值为 1 或更大。默认值：1。
BitsPerComponent	整型	（仅当 Predictor 大于 1 时使用）用于表示样本中每个颜色分量的位数.有效值为 1，2,4，8,（在 PDF 1.5）16.默认值：8.
Columns	整型	（仅当 Predictor 大于 1 时使用）每行中的样本数。默认值：1。
EarlyChange	整型	（仅限 LZWDecode）指示何时增加编码长度.如果此条目的值为 0，则编码长度的增加将尽可能延长。如果值为 1，则编码长度增加会发生一个编码。包含这个参数是因为一些供应商分发的 LZW 示例编码提高了一段编码的编码长度。默认值：1。

LZW 算法和 Flate 算法的预测函数

3.3。4 RunLengthDecode 过滤器

3.3。5 CCITTFaxDecode 过滤器

3.3.6 JBIG2Decode 过滤器

3.3。7 DCTDecode 过滤器

3。3.8 JPXDecode 过滤器

3.3.9 Crypt 过滤器

3.4 文件结构

前面的部分描述单个对象的语法。本节介绍如何组织 PDF 文件中的对象进行有效的随机访问和增量更新。 规范的 PDF 文件最初由四个元素组成（见图 3.2)：

- 头部(header），一个单行的头，标识文件符合的 PDF 规范的版本。
- 正文(body），包含构成文件内文档内容的对象。
- 交叉引用表（cross—reference table），包含文件内间接对象的信息.
- 尾部（trailer），提供交叉引用表的位置以及文件正文内的某些特殊对象。

这个初始结构可以被后面的更新所修改,该更新在文件的末尾增加附加元素；详见 3.4.5 节“增量更新”。

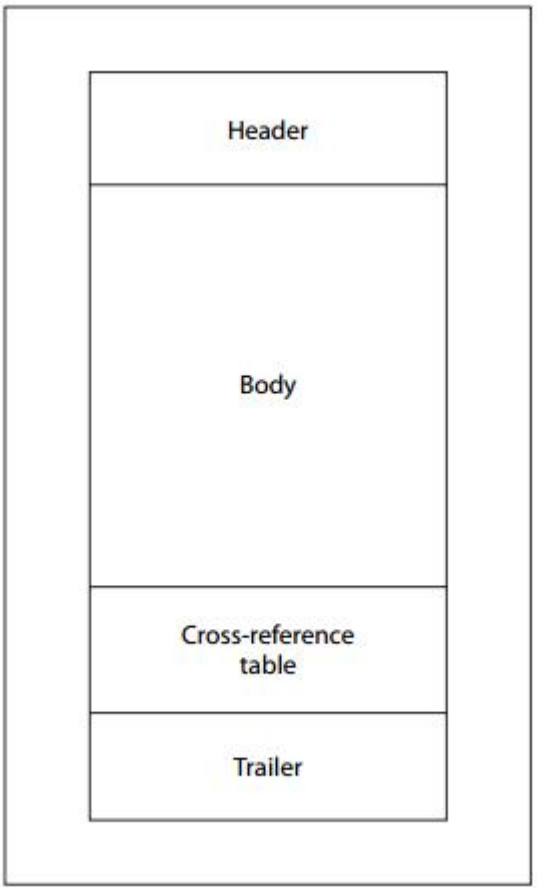

图 3.2 PDF 文件的初始结构

按照惯例，PDF 文件中的标志排列成行；见第 3。1 节“词汇约定”.每一行都由行尾（EOL）结束符终止，可能是回车符(字符代码 13），换行符(字符代码 10)或回车换行符.具有二进制数据的 PDF 文件可能有任意长度的行。但是,为了增加与其他处理 PDF 文件应用程序的兼容性，行，如果不是流对象数据的一部分，其长度被限制为不超过 255 个字符。但有一个例外，从 PDF 1.3 开始,签名字典的内容字符串（见 8。7 节“数字签名”）不受行长度限制。另见附录 H.

这里描述的规则足以产生一个格式良好的 PDF 文件。但是，附加规则适用于组织 PDF 文件，以便能够在网络环境中高效地增量访问文档的组件.这种组织形式称为线性化 PDF，在附录 F 中有所描述。

3。4.1 头部

PDF 文件的第一行是标识文件符合的 PDF 规范的版本。对于符合 PDF 1.7 规范的文件，标题

应该是

%PDF-1。7

但是，由于任何符合早期版本 PDF 规范的文件也符合 1.7 版本,处理 PDF 1.7 的应用程序也可以接受使用以下任何标题的文件：

%PDF-1.0

%PDF-1.1

%PDF-1。2

%PDF-1.3

%PDF-1.4

%PDF-1。5

%PDF-1。6

(另见附录 H 中的实施说明 13 和 14)

从 PDF 1。4 开始,文件头部中的版本可以被文档目录字典中的 **Version** 条目覆盖(通过尾部中的 **Root** 条目定位，如第 3.4.4 节“尾部"中所述）。这使 PDF 生产者应用程序可以使用增量更新来更新版本（请参见第 3.4.5 节“增量更新"）。

在某些情况下，PDF 文件的版本比设计时接受的版本更新，消费者应用程序也应能够处理这些 PDF 文件。新的 PDF 功能通常以这种方式引入，如果消费者应用程序不能理解它们，则它们会被安全地忽略（见第 H.1 节“PDF 版本号”）。

注意：如果 PDF 文件包含二进制数据,就像大多数的 PDF 文件(参见第 3。1 节“词汇约定”），建议头部行后面紧跟一条至少包含四个二进制字符的注释行，即字符的编码为 128 或更大。这确保文件传输应用程序的正确行为，用于检查文件开头附近的数据，以确定是否将文件的内容视为文本或二进制文件。

3.4。2 正文

PDF 文件的正文由一系列表示文档内容的间接对象组成.对象，其基本类型的描述在第 3。2 节“对象”中，表示文档的组件,例如字体，页面和采样图像。从 PDF 1.5 开始,正文还可以包含对象流，每个对象流都包含一系列间接对象；请参见第 3。4.6 节“对象流”。

3.4。3 交叉引用表

交叉引用表包含允许随机访问文件中的间接对象的信息，以便不需要读取整个文件来定位特定对象.该表包含每个间接对象的单行条目，指定该对象在文件中的位置。(从 PDF 1。5 开始，部分或全部交叉引用信息可以替代地包含在交叉引用流中;请参见第 3。4。7 节“交叉引用流”)。

交叉引用表允许随机访问表中的条目，在 PDF 文件中唯一并且具有固定的格式。该表包括一个或多个交叉引用段.最初，整个表由单个段组成（如果文件是线性化的,则包含两个段；见附录 F）。每次更新文件时都会添加一个段（请参见第 3。4.5 节“增量更新”）。

每个交叉引用段以包含关键字 xref 的行开头。开头行之后是一个或多个交叉引用子段,可以以任意顺序出现。子段结构对增量更新非常有用，因为它允许将新的交叉引用段添加到 PDF 文件中，其中仅包含已添加或删除的对象的条目。对于从未更新的文件，交叉引用段只包含一个小节,其对象编号从 0 开始。

每个交叉引用子段包含连续范围的对象编号的条目.子段的开始行，是以空格间隔的两个数字:首对象的编号和条目的数量。例如，行

28　5

引入一个包含五个对象的子段，对象编号从 28 到 32。

注意:在单个段中，对象编号不能在多个子段中具有条目。但是，请参见附录 H.

以下这行是交叉引用条目本身，每行一个。每个条目正好是 20 个字节长，包括行尾标记。有两种交叉引用条目：一种用于正在使用的对象，另一种用于已被删除的对象（所以是空闲的）。

两种类型的条目具有相似的基本格式，由关键字 n（对于使用中的条目）或 f（对于空闲条目）来区分。使用条目的格式是

nnnnnnnnnn ggggg n eol

其中

nnnnnnnnnn 是一个 10 位的字节偏移

ggggg 是一个 5 位的生成编号

n 是一个关键字，将其标识为使用中的条目

eol 是一个 2 个字符的行尾序列

字节偏移是一个 10 位数字（不够 10 位的数字，前面填充零），给出从文件开始到对象开头的字节数,与生成编号之间间隔一个空格。生成编号是一个 5 位数字（不够 5 位的数字,前面填充零 ）。生成编号后面是单个空格，以及关键字 n 和 2 个字符的行尾序列。如果文件的行尾标记是单个字符（回车符或换行符），则前面加上一个空格；如果标记为 2 个字符（回车符和换行符），则不加空格。因此，条目的总长度始终为 20 个字节.

空闲对象的交叉引用条目具有基本相同的格式，除了关键字是 f 而不是 n，第一个项的解释是不同的：

nnnnnnnnnn ggggg f eol

其中

nnnnnnnnnn 是下一个空闲对象的 10 位对象编号

ggggg 是一个 5 位生成编号

f 是一个关键字，将其标识为空闲条目

eol 是一个 2 个字符的行尾序列

交叉引用表中的空闲条目形成链表,每个空闲条目包含下一个的对象编号.表中的第一个条目（对象编号 0）始终是空闲的,并且生成编号是 65,535。它是空闲对象链表的头。最后一个空闲条目（链表的尾部）链接回编号 0 对象。（此外，表可能包含其他的链接回编号 0 对象并且

生成编号为 65,535 的空闲条目,即使这些条目是不在链表中。）见附录 H。中的实施说明 16。

除了编号 0 对象之外，交叉引用表中的所有对象最初的生成编号都是 0。当间接对象被删除时，其交叉引用条目被标记为空闲的，并将其添加到空闲条目的链表中。条目的生成编号增加 1，以指示下一次创建具有该对象编号的对象时，使用的生成编号。因此，每次重复使用交叉引用条目时，都会给出一个新的生成编号.生成编号的最大值是 65,535； 当达到该值时，交叉引用条目不会再被重复使用.

交叉引用表（包括原始交叉引用段和所有更新段）必须为每个对象编号（范围从 0 到文件中使用的最大对象编号）包含一个条目，即使上述范围中的一个或多个对象编号实际上并没有出现在文件中。见附录 H 中的实施说明 17。

示例 3.5 显示了由具有六个条目的子段组成的交叉引用段：正在使用的四个(对象编号 1，2，4 和 5）和两个空闲（对象编号 0 和 3）。对象编号 3 已被删除，并且使用该对象编号创建的下一个对象的生成编号为 7.

示例 3。5

```
xref
0 6
0000000003 65535 f
0000000017 00000 n
0000000081 00000 n
0000000000 00007 f
0000000331 00000 n
0000000409 00000 n
```

示例 3.6 显示了具有四个子段的交叉引用段，共包含五个条目。第一个子段包含一个条目，对象号为 0，它是空闲的。第二个子段包含一个条目，对象编号 3，正在使用.第三个子段包含两个条目，对象编号分别是 23 和 24,它们都在使用中。对象编号 23 已被重复使用,实际上，可

以看出，它的生成编号是 2.第四个子段包含一个条目，对象编号 30,正在使用中。

示例 3.6

```
xref
0 1
0000000000 65535 f
3 1
0000025325 00000 n
23 2
0000025518 00002 n
0000025635 00000 n
30 1
0000025777 00000 n
```

有关更多的已更新多次的 PDF 文件结构的示例，请参见第 G.6 节“更新示例”.

3.4。4 尾部

PDF 文件的尾部使读取文件的应用程序能够快速查找交叉引用表和某些特殊对象.应用程序应从其尾部读取 PDF 文件。文件的最后一行只包含文件结尾标记，%%EOF(见附录 H 中的实现说明 18）。最后一行的前两行，包含 **startxref** 关键字和文件开头到最后交叉引用段的 **xref** 关键字前的字节偏移量。**startxref** 行前面是尾部字典，包括关键字 **trailer**，后跟一系列包含在双尖括号（〈< ..。 >〉）中的键值对.所以,尾部具有以下整体结构：

```
trailer
  << key1 value1
    key2 value2
    …
```

keyn valuen

>>

startxref

Byte_offset_of_last_cross-reference_section

%%EOF

表 3.13 列出了尾部字典的内容。

表 3。13 尾部字典条目

键值	类型	说明
Size	整形	（必需;不能是间接引用）文件交叉引用表中的条目总数,由原始部分和所有更新部分一起确定。同样地,该值比文件中使用的最大对象编号大 1。*注意:数字大于此值的交叉引用段中的任何对象都将被忽略,并被视为缺失。*
Prev	整形	（仅当文件具有多个交叉引用段时才存在;不能是间接引用）从文件开始到前一个交叉引用段开头的字节偏移量。
Root	字典	（必需;必须是间接引用）文件中 PDF 文档的目录字典(参见第 3.6.1 节“文档目录”）。
Encrypt	字典	（如果文档被加密，则为必需； PDF 1.1）文档的加密字典(请参见第 3.5 节“加密”）。
Info	字典	（可选;必须是间接引用）文档的信息字典（参见第 10.2。1 节“文档信息字典”）。
ID	数组	（可选，但强烈推荐； PDF 1.1)一个数组，包含两个字节字符串，构成文件的文件标识符（参见第 10。3 节“文

件标识符”）。两个字节字符串应该是直接对象，并且是不加密的。虽然此条目是可选的，但它的缺失可能会影响依赖于唯一标识文件的某些工作流中的功能.

*注意：表 3。17 定义了一个额外的条目,**XRefStm**, 仅在混合引用文件的尾部出现.参见 3。4。7 节中的“不支持 PDF1.5 的应用程序的兼容性”.*

示例 3。7 显示了一个从未更新过的文件的尾部（即尾部字典中没有 **Prev** 条目）。

示例 3。7

```
trailer
   <〈 /Size 22
     /Root 2 0 R
     /Info 1 0 R
     /ID  [  〈81b14aafa313db63dbd6f981e49f94f4〉
          〈81b14aafa313db63dbd6f981e49f94f4〉
        ]
   〉>
startxref
18799
%%EOF
```

3.4。5 增量更新

PDF 文件的内容可以增量更新,无需重写整个文件。更改附加到文件的末尾,保留其原始内容.以这种方式更新文件的主要优点（如第 2.2.7 节“增量更新”中所述），可以快速保存大型文档的小小更改。还有其他优点:

- 在某些情况下，增量更新是保存文档更改的唯一方法。在保存文档时，一种被接受的最小化数据丢失风险的做法是将其写入新文件，并重新命名新文件以替换旧文件。但是，在某些情况下，例如通过 HTTP 连接编辑文档或使用 OLE 嵌入（Windows 特定技术）时，不可能以这种方式覆盖原始文件的内容. 增量更新可用于保存对这些情况下文档的修改。
- 一旦文档签名（见第 2.2.6 节“安全性”），对文档所做的所有更改都必须使用增量更新进行保存，因为更改文件中的任何现有字节都将使现有签名无效。

在增量更新中，任何新的或已更改的对象都将附加到文件中，并添加交叉引用段，并插入一个新的尾部。生成的文件具有如图 3。3 所示的结构。更新文件的完整示例如第 G.6 节“更新示例"所示。

当文件更新时添加的交叉引用段仅包含已更改，替换或删除的对象的条目. 删除的对象在文件中保持不变，但通过交叉引用条目将其标记为已删除。添加的尾部包含来自前一个尾部的所有条目（可能已修改），以及前一个交叉引用段的位置的上一个条目（请参见 3。4。4 节中的表 3.13）.如图 3。3 所示,已经多次更新的文件包含几个尾部；每个尾部都由其自己的文件结尾（%% EOF)标记终止。

由于更新附加到 PDF 文件，文件可以具有相同对象标识符（对象编号和生成编号)的对象的多个副本。例如，如果文本注释（参见第 8.4 节“注释”）被更改多次,并且文件在更改时都保存，则可能会发生这种情况。因为文本注释对象不被删除，它保留了与之前相同的对象编号和生成编号.对象的更新副本包含在添加到该文件的新的更新部分中。更新的交叉引用段包括对该对象的新副本的字节偏移，覆盖原始交叉引用段中包含的旧字节偏移量.当消费者应用程序读取文件时，它必须以这样的方式构建其交叉引用信息，即每个对象的最新副本是文件中需要访问的对象。

在 1。4 之前的 PDF 版本中，由于版本仅在文件头部中指定,所以无法使用增量更新来更改文档符合的 PDF 版本（请参见第 3。4。1 节 ，“头部”）.在 PDF 1。4 中，文档目录字典中的 Version 条目（参见第 3.6.1 节“文档目录”）可以覆盖头部中指定的版本，从而可以使用增量更新来

更改版本。

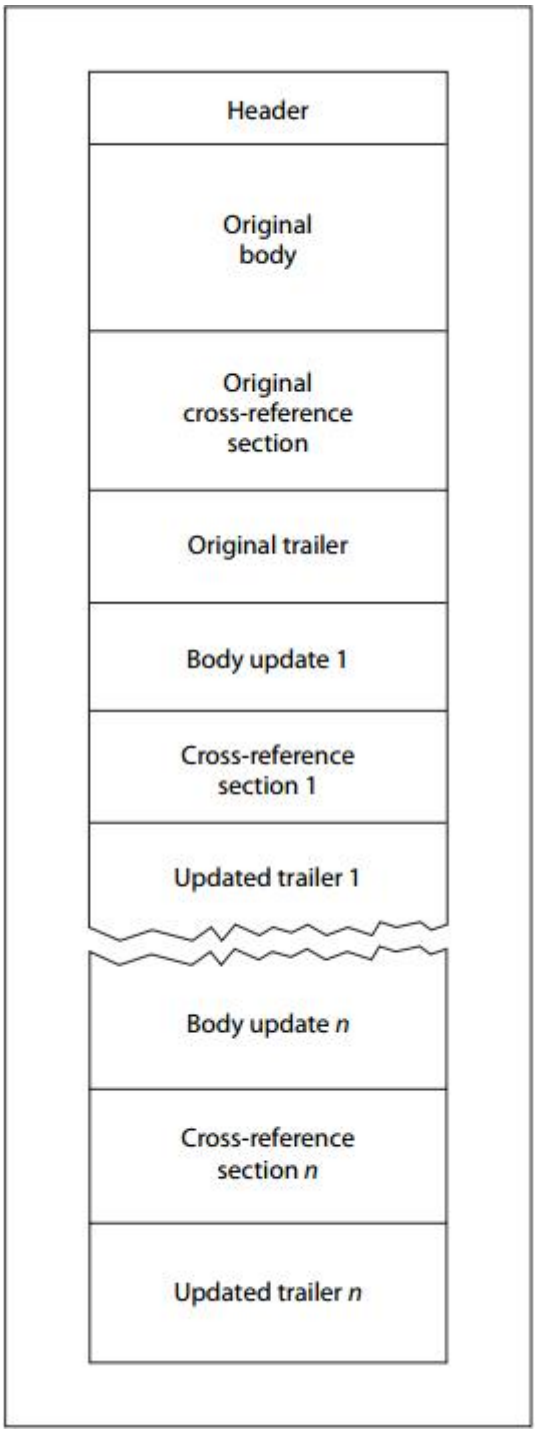

图 3。3 更新过的 PDF 文件的结构

3。4。6 对象流

PDF 1。5 引入了一种新的流，一个包含一系列 PDF 对象的对象流。对象流的目的是允许更多数量的 PDF 对象被压缩，从而大大减少 PDF 文件的大小。流中的对象被称为压缩对象。(无论流

是实际上使用过压缩过滤器进行编码，都使用该术语。）

任何 PDF 对象都可以出现在对象流中，但有以下例外：

- 流对象
- 生成编号为零的对象
- 文档的加密字典（参见第 3。5 节“加密”）
- 表示对象流字典中 Length 条目值的对象

注意:另外，在线性化文件(见附录 F“线性化 PDF”)中，文档目录，线性化字典和页面对象可能不会出现在对象流中.

间接引用对象流中的对象使用正常的语法：例如，14 0 R.访问这些对象需要一个不同的方法去存储交叉引用信息。请参见第 3。4.7 节“交叉引用流”。虽然应用程序因为使用压缩对象，必须支持 PDF 1.5，但也可以用和 PDF 1。4 兼容的方式存储对象。不支持 PDF 1.5 的应用程序可以忽略对象； 请参阅 3。4.7 节中的“不支持 PDF 1。5 的应用程序的兼容性”。

除了表 3.4 所示的流标准键值之外，描述对象流的流字典包含以下条目：

表 3。14 对象流字典特有的附加条目

键值	类型	描述
Type	名字	（必需）此字典描述的 PDF 对象的类型； 对象流必须是 **ObjStm**.
N	整型	(必需)流中压缩对象的数量。
First	整型	(必需)第一个压缩对象的字节偏移（在解码流中）.
Extends	流	（可选）对对象流的引用，此引用也是对象流，并被视为扩展。两个流都被视为对象流集合的一部分（见下文）。给定的集合由一组流组成，这些流的 **Extends** 链接构成一个定向的非循环图。

PDF 文件的创建者可以灵活地确定哪些对象（如果有的话）存储在对象流中. 例如，将具有共同特征的对象存储在一起是有用的,例如“第 1 页上的字体”或“草稿＃3 的注释”。这些存储在一起的对象被称为集合.

为了避免诸如在下载和解压缩大对象流以访问单个压缩对象时发生的性能下降问题，应当限制单个对象流中的对象数量.（见附录 H 中的实现说明 19）这可能需要一组对象流作为集合进行链接，这可以通过对象流字典中的 **Extends** 条目来完成。

当更新集合以包含新对象时，也可以使用 **Extends**。不需要重新定义原始对象流，这将需要复制流数据,新对象可以存储在一个新的对象流中.在文档中添加更新部分时,这一点尤为重要。对象流中的流数据由以下项组成：

- N 对整数，其中每对中的第一个整数表示压缩对象的对象编号，第二个整数表示该对象相对于第一个整数的字节偏移量.偏移量必须按照增加的顺序排列，但对对象编号的顺序没有限制.

  *注意：解码流中第一个对象的字节偏移,是 **First** 条目的值。*

- N 个对象连续存储。只有对象值存储在流中；不使用 **obj** 和 **endobj** 关键字。 一个压缩字典或数组可能包含间接引用。

  注意：压缩对象只包含间接引用是非法的； 例如,3 0 R。

相比之下，内容流中的字典和数组（第 3.7。1 节）可能不包含间接引用。在加密文件中，由于整个对象流被加密，所以在对象流中任何地方的字符串不能被单独加密。

注意：第一个对象的数据不需要立即跟随最后一个字节偏移量。未来的扩展可能会在流中的这两个点之间放置附加信息。

像任何流一样,对象流本身就是一个间接对象,它必须在一个交叉引用表或交叉引用流中（参见第 3.4.7 节“交叉引用流”），尽管可能没有任何引用（形式为 243 0 R）.

对象流和任何压缩对象的生成编号隐式地为零。如果对象流或压缩对象被删除并且对象编号

被释放，那么该对象编号只能用于除对象流之外的普通(未压缩）对象。当创建新的对象流和压缩对象时，必须分配新的对象编号,而不是从空闲列表中取出旧的对象编号。

示例 3。8 显示了三个对象（两个字体和一个字体描述符），它们在 PDF 1.4 或更早的文件中，随着交叉引用表一起表示。在示例 3.9 中，相同的对象在 PDF 1。5 文件中,随着交叉引用流一起，存储在对象流中。

示例 3.8

```
11 0 obj
   〈< /Type /Font
    /Subtype /TrueType
    。..other entries。..
    /FontDescriptor 12 0 R
  >>
endobj
12 0 obj
  <〈 /Type /FontDescriptor
    /Ascent 891
     。。。other entries。。。
    /FontFile2 22 0 R
  〉>
endobj
13 0 obj
  <〈 /Type /Font
    /Subtype /Type0
    .。。other entries.。.
```

```
    /ToUnicode 10 0 R
  >>
endobj
。。。
xref
0 32
0000000000 65535 f
。..
0000001434 00000 n % 对象 11 的交叉引用条目
0000001735 00000 n % 对象 12 的交叉引用条目
0000002155 00000 n % 对象 13 的交叉引用条目
。..
trailer
  〈〈 /Size 32
    /Root .。.
  >>
```

在示例 3.9 中,交叉引用流（参见第 3.4.7 节“交叉引用流”）包含字体条目(对象 11 和 13)和字体描述符条目(对象 12)，它们是对象流中的压缩对象.这些条目的第一个字段是条目类型(2)，第二个字段是对象流（15）的编号，第三个字段是对象流中的对象序列（0，1 和 2）。交叉引用流还包含对象流本身类型为 1 的条目.

注意：为了可读性,对象流使用未编码的形式显示。在一个真正的 PDF 1.5 文件中，通常会使用 Flate 编码进行压缩。

示例 3。9

```
15 0 obj     % 流对象
```

```
   << /Type /ObjStm
      /Length 1856
      /N 3       % 流中对象的数量
      /First 24     % 第一个对象的字节偏移
   >>
stream
   % 对象的对象编号和相对于第一个对象的偏移量
   11 0 12 547 13 665
   << /Type /Font
      /Subtype /TrueType
      ...other keys...
      /FontDescriptor 12 0 R
   >>
    << /Type /FontDescriptor
      /Ascent 891
      .。。other keys。..
      /FontFile2 22 0 R
   >>
   << /Type /Font
      /Subtype /Type0
      ..。other keys..。
      /ToUnicode 10 0 R
   >>
..。
```

```
endstream
endobj
99 0 obj  % 交叉引用流
    〈〈 /Type /XRef
      /Index  [0 32]       % 这个段有一个子段，拥有 32 个对象
      /W  [1 2 2]  % 每个条目都有三个字段：字节宽分别是 1，2 和 2
      /Filter /ASCIIHexDecode    % 为了此例中的可读性,使用此编码器
      /Size 32
      。。。
   >>
stream
   00 0000 FFFF     % 在交叉引用表中的“0 65535 f”
   .。。
   02 000F 0000     % 对象 11 的条目,第一个字体对象。
   02 000F 0001     % 对象 12 的条目,字体描述对象。
   02 000F 0002 % 对象 13 的条目，第二个字体对象。
   。.。
   01 BA5E 0000     % 对象 15 的条目,流对象.
   。..
endstream
endobj
startxref
   54321  % “99 0 obj”的字节偏移
%%EOF
```

3。4.7 交叉引用流

从 PDF 1。5 开始，交叉引用信息可能存储在交叉引用流中，而不是交叉引用表中。交叉引用流具有以下优点：

- 更紧凑的表示交叉引用信息。
- 访问存储在对象流中压缩对象的能力（请参见第 3。4。6 节“对象流”),并允许将来添加新的交叉引用条目类型.

交叉引用流是流对象（参见第 3。2.7 节“流对象”）,包含字典和数据流。每个交叉引用流包含与交叉引用表(参见第 3。4.3 节“交叉引用表”）相当的信息和一个交叉引用部分的尾部（参见第 3.4.4 节“尾部”）.尾部字典条目存储在流字典中，交叉引用表条目存储为流数据，如以下示例所示：

示例 3。10

```
..。 objects 。..
12 0 obj  % 交叉引用流对象
    〈〈 /Type /XRef    % 交叉引用流字典
      /Size ..。
      /Root 。。.
    〉〉
stream
。..    % 包含交叉引用信息的流数据
endstream
endobj
。.. more objects .。。
startxref
```

byte_offset_of_cross-reference_stream % 指向对象 12

%%EOF

请注意，**startxref** 关键字后面的值现在是交叉引用流而不是 xref 关键字的偏移.(见附录 H 中的实现注释 21）对于完全使用交叉引用流的文件（即不是混合引用的 PDF 1。5 文件），请参见 3。4.7 节中的“不支持 PDF 1。5 的应用程序的兼容性”，关键字 **xref** 和 **trailer** 不再使用.因此，除了 **startxref**、**%%EOF** 段和注释外，PDF 1。5 文件完全是对象的序列.

注意：在线性化 PDF 中允许使用对象流和交叉引用流，对规范进行微小的修改（参见 F.2 节“线性化 PDF 文档结构"）。

交叉引用流字典

除了所有流（表 3。4）和尾部字典（表 3。13）共有的条目之外，交叉引用流还包含表 3。15 所示的条目。由于消费者应用程序需要交叉引用流中的某些信息来构建允许间接引用的索引，所以交叉引用流中的条目受以下限制：

- 表 3.15 所示的所有条目的值必须是直接对象；不允许间接引用。对于数组(**Index** 和 **W** 条目)，它们的所有元素也必须是直接对象。如果流被编码，表 3。4 中的 **Filter** 和 **DecodeParms** 条目也必须是直接对象。另见附录 H 中的实施说明 20。

  *注意:表 3.15 中未列出的其他交叉引用流条目可能是间接的；实际上,有些（如表 3。13 中的 **Root**）需要是间接的。*

- 交叉引用流不得被加密,也不得在交叉引用流字典中出现任何字符串。它不能有指定 **Crypt** 过滤器的 **Filter** 条目（参见 3。3。9“Crypt 过滤器”）。

表 3。15 交叉引用流字典的特定额外条目

键值	类型	描述
Type	名字	（必需）此字典描述的 PDF 对象的类型；交叉引用流必须是

XRef。

Size	整型	（必需）一个数字，比此段或此段更新后的段使用的最高的对象编号大 1。它等同于尾部字典中的 Size 条目.
Index	数组	（可选）一个数组，包含给此段中的每个子段使用的整数对. 第一个整数是子段中的第一个对象的编号，第二个整数是子段中条目的数量。 数组按对象编号升序排列。子段不能重叠； 一个对象号在一个子段中最多有一个条目。 默认值：[0 Size]
Prev	整型	（仅当文件具有多个交叉引用流时存在;在混合引用文件中无意义;请参见 3。4。7 节中的“不支持 PDF 1。5 的应用程序的兼容性")从文件开头到上一个交叉引用流的开始。 该条目与尾部字典中的 Prev 条目具有相同的功能（表 3.13）.（另见附录 H 中的实施说明 21）
W	数组	（必需）表示单个交叉引用条目中字段大小的整数数组。表 3.16 描述了条目类型及其字段.对于 PDF 1.5，W 总是包含三个整数； 每个整数的值是相应字段的字节数(在解码流中）。例如，[1 2 1] 表示字段分别是一个字节，两个字节和一个字节。 W 数组中的元素的值为零表示流中不存在相应的字段，如果有的话，则使用默认值.如果第一个元素为零，则类型字段不存在，并且默认为类型 1。 各项的总和是每个条目的总长度； 它可以与 Index 数组一起使用，以确定每个子段的起始位置。

注意:PDF 文件中的不同交叉引用流可能对 W 使用不同的值.

交叉引用流数据

交叉引用流中的每个条目都有一个或多个字段,第一个字段指定条目的类型(参见表 3。16).在 PDF 1。5 中,仅允许类型 0，1 和 2。任何其他值被解释为对 null 对象的引用，从而允许将来定义新的条目类型。

这些字段以字段编号的增加顺序编写；每个字段的长度由 W 条目中的相应值确定(见表 3.15）。需要多于一个字节的字段首先与高位字节一起存储.

表 3.16 交叉引用流条目

类型	字段	描述
0	1	条目的类型必需是 0.类型 0 条目定义空闲对象的链接列表（对应于交叉引用表中的 f 条目）。
	2	下一个空闲对象的对象编号。
	3	生成编号,如果此对象编号重新使用的话。
1	1	条目的类型必需是 1.类型 1 条目定义对象在使用但没有压缩（对应于交叉引用表中的 n 条目）。
	2	对象的字节偏移，从文件头开始。
	3	对象的生成编号，默认值:0。
2	1	条目的类型必需是 2.类型 2 条目定义压缩的对象。
	2	存储此对象的流对象的对象编号(流对象的生成编号隐式为 0)。
	3	流对象中此对象的索引号。

像任何流对象一样，交叉引用流是间接对象。因此，它的条目必须存在于交叉引用流（通常本身)或交叉引用表（混合引用文件中）中;请参见 3。4。7 节中的“不支持 PDF 1.5 的应用程序的兼容性”)。

不支持 PDF 1。5 的应用程序的兼容性

不支持 PDF 1。5 的应用程序无法访问交叉引用流引用的对象。如果文件仅使用交叉引用流,则不能由这些应用程序打开.

但是，可以构造一个混合引用文件,该文件可被 PDF 1.4 消费者应用程序读取。除了被交叉引用流引用的对象流中的对象之外，这样的文件还包含由标准交叉引用表引用的对象.

在这些文件中，除了表 3。13 所示的尾部条目外，尾部字典还可以包含一个附加条目，如表 3.17 所示.PDF 1。4 消费者忽略此条目，因此，无法访问该条目所引用的交叉引用流中的条目.

表 3。17 混合引用尾部字典的附加条目

键值	类型	说明
XRefStm	整型	从文件头到交叉引用流之间的字节偏移。

尾部的 **Size** 条目必须足够大以包括所有对象，包括在 **XRefStm** 条目引用的交叉引用流中定义的对象.但是，为了允许随机访问，主交叉引用段必须包含编号为 0 到 Size-1 的所有对象的条目（参见第 3。4。3 节“交叉引用表”）。 因此，**XRefStm** 条目不能在主交叉引用段的尾部字典中使用，而只能在更新交叉引用段中使用。

当 PDF 1。5 消费者打开混合引用文件时,不会隐藏具有交叉引用流中条目的对象。当应用程序搜索对象时，如果在任何给定的标准交叉引用段中找不到条目，则搜索进入 **XRefStm** 条目指定的交叉引用流，然后查看前一个交叉引用段（**Prev** 条目在尾部）。

因此，隐藏对象有两个交叉引用条目。一个是在交叉引用流中。另一个上一段中的一个空闲条目,通常是由 **Prev** 条目引用的部分。PDF 1。5 消费者首先查看交叉引用流，找到该对象,并忽

略上一段中的空闲条目。PDF 1.4 消费者忽略交叉引用流,并在上一段中查找空闲条目。空闲条目必须具有 65535 的下一个生成编号,以使对象编号不会被重复使用。

混合引用文件中的哪些对象可以被隐藏是有限制的，从而不会使文件对 PDF 1.4 和更早版本的消费者无效。特别是，PDF 文件的根目录（见第 3.6.1 节“文档目录"）不能被隐藏，也不能从根目录中看到任何对象.这些对象可以从根开始并递归地确定:

- 在任何可见字典中,直接对象都可见。任何所需键值对的值是可见的。
- 在任何可见数组中，每个元素都可见。
- 内容流中的资源字典是可见的。虽然资源字典不是必需的，但严格来说，它依附的内容流被假定包含对资源的引用。

一般来说,可隐藏的对象是由间接引用指定的可选对象。PDF1.5 消费者可以通过交叉引用流来解析这些引用。而对于 PDF1.4 消费者，对象被视为是空闲对象，引用被视为是对空对象的引用。

例如，**Outlines** 条目在目录中是可选的。因此，其值可能是对隐藏对象的间接引用.PDF1.4 消费者将其视为对空对象的引用，相当于完全忽略了条目,而 PDF1。5 消费者可以识别.但是，如果 **Outlines** 条目的值是对可见对象的引用，则整个大纲树必需可见，因为大纲树节点包含指向其他节点的所需指针.

遵循此逻辑，必需可见的项包括整个页面树，字体，字体描述符和宽度表。在混合引用文件中可能被隐藏的对象包括结构树，大纲树，文章线，注释，目标地址，网络捕获信息和页面标签。

示例 3.11 显示了包含主交叉引用段和更新交叉引用段的混合引用文件，其是 XRefStm 条目指向交叉引用流（对象 11)，后者又引用对象流（对象 2）。

在此示例中，目录（对象 1）包含对结构树根目录的间接引用（3 0 R)。对象的搜索从更新交叉引用表开始，但表中没有对象.搜索的继续，取决于消费者应用程序的版本:

- 对于 PDF1.4 消费者，继续按照指向主交叉引用表的 **Prev** 条目进行搜索.该表将对象 3 定义

为空闲对象,也就是将其视为空对象。因此，PDF1.4 消费者认为此条目是缺失的，并认为该文档没有结构树。

- 对于 PDF1。5 消费者，继续按照指向交叉引用流（对象 11）的 **XRefStm** 条目进行搜索。该流将对象 3 定义为压缩对象，存储位置是对象流（2 0 obj)的索引 0.因此，PDF1.5 消费者认为该文档有结构树。

注意：为了此示例中交叉引用流格式和内容的可读性，指定了 ASCIIHexDecode 过滤器。如附录 H 中的实施说明 20 所述，该示例将不会被 Acrobat6.0 和更高版本的查看器所接受。

示例 3.11

```
1 0 obj        %文档根对象，偏移值 23
    〈< /Type /Catalog
      /StructTreeRoot 3 0 R
      …
    >〉
endobj
12 0 obj
…
endobj
…
99 0 obj
…
endobj
xref           %主交叉引用段，偏移值 2664
0 100      %这个子段拥有对象 0-99 的条目。
0000000002 65535 f      %对象 0 条目
```

```
0000000023 00000 n      %对象 1 条目,根对象
0000000003 65535 f      %对象 2（对象流）条目,此表中标记为空闲
0000000004 65535 f      %对象 3 条目，此表中标记为空闲
0000000005 65535 f      %。。。
0000000006 65535 f
0000000007 65535 f
0000000008 65535 f
0000000009 65535 f
0000000010 65535 f
0000000011 65535 f
0000000000 65535 f      %对象 11(交叉引用流)条目，此表中标记为空闲。
0000000045 00000 n      %对象 12 条目，使用中.
0000000179 00000 n      %对象 13 条目，使用中。
…
0000002201 00000 n      %对象 99 条目,使用中。
trailer
   << /size 100
      /Root 1 0 R
      /ID …
   >>
startxref
   2664       %主交叉引用段的偏移值
%%EOF
2 0 obj        %对象流，偏移值 3722
```

```
    〈〈 /Length …
      /N 8              %此流包含 8 个对象。
      /First 47         %第一个对象相对此流的偏移(流偏移)
    >>
stream
    3 0 4 50 5 72 …     %8 个对象的对象编号和流偏移（相对于第一个对象）
    〈〈 /Type /StructTreeRoot        %对象 3
      /K 4 0 R
      /RoleMap 5 0 R
      /ClassMap 6 0 R
      /ParentTree 7 0 R
      /ParentTreeNextKey 8
    >>
    << /S /Workbook          %对象 4（K 值从 StructTreeRoot 获取）
      /P 8 0 R
      /K 9 0 R
    >>
    << /Workbook /Div        %对象 5（RoleMap）
      /Worksheet /Sect
      /TextBox /Figure
      /Shape /Figure
    >〉
    …          %对象 6 至 10 在这里定义
endstream
```

```
endobj
11 0 obj          %交叉引用流，偏移值 4899
   << /Type /XRef
      /Index [2 10]    %这个流包含对象 2 至 11 的条目
      /Size 100
      /W [1 2 1]       %每个字段的字节宽
      /Filter /ASCIIHexDecode%仅为可读性（Acrobat6 不支持）
   >>
stream
   01 0E8A 0
   02 0002 00       %对象 3（对象流 2,索引 0）条目
   02 0002 01       %对象 4（对象流 2，索引 1)条目
   02 0002 02       %。。。
   02 0002 03
   02 0002 04
   02 0002 05
   02 0002 06
   02 0002 07       %对象 10（对象流 2，索引 7）条目
   01 1323 0     %对象 11 条目(0x1323=4899)
endstream
endobj
xref
0 0
trailer
```

```
    << /Size 100
       /Prev 2664
       /XRefStm 4899
       /Root 1 0 R
       /ID …
    >>
startxref
    5640
%%EOF
```

该示例说明了其他几点:

- 对象流是未编码的,交叉引用流使用 ASCII 十六进制编码.实际上,两个流都会使用 Flate 编码。此外，示例中的交叉引用表中显示的注释仅用于说明目的，在交叉引用表中，PDF 注释是不合法的。
- 隐藏的对象，2 至 11，对象编号是连续的.实际上，没有这样的要求，也没有要求交叉引用表中的空闲项的编号以升序连接直到最后.
- 更新交叉引用表不包含条目,这不是必需的，但是是合理的。使用混合引用格式的 PDF 创建者，会同时创建主交叉引用表,更新交叉引用表和交叉引用流.例如，对象 12 和 13 没有被压缩。它们可能在更新表中有条目。由于对象 2 和 11,对象流和交叉引用流没有被压缩，它们同样可能在更新表中被定义。然而，它们是隐藏段的一部分，所以在交叉引用流中定义它们是有意义的。
- 更新交叉引用段必需出现在文件末尾,除此之外,任何对象或主交叉引用段都没有排序限制。但是，使用混合引用格式和线性化格式的文件，都有排序限制。

3。5 加密

PDF 文档可以被加密(PDF1.1),以保护其内容不被未经授权的访问.加密适用于 PDF 文件的所有文档中的字符串和流，但不适用于其它对象类型,比如整型和布尔,这些对象主要用于传递文档结构相关的信息而不是其内容的信息。未加密允许随机访问文档中的对象，而加密字符串和流保护文档的实质内容。

注意：当一个 PDF 流对象（见 3.2.7 节，“流对象”）引用一个外部文件，这个流的内容是未加密的，因为外部文件不是 PDF 文件本身的一部分.但如果流的内容嵌入到 PDF 文件中（见 3。10.3 节，“嵌入式文件流”），它们就会像文件中其它流一样被加密.从 PDF1。5 开始，嵌入式文件可以在未加密的文件中加密(见 3。5。4 节，“加密过滤器”)。

加密的有关信息存储在加密字典中,该字典是文档尾部字典中 **Encrypt** 条目的值(见 3.4。4 节中的表 3.13）。尾部字典中没有这个条目，那意味着文档没有被加密。表 3.18 所示的条目对于所有加密字典都是通用的。

加密字典中的 **Filter** 条目标识了文件的安全处理器,它是一个软件模块,实现加密过程中的各个方面，并控制对加密文档内容的访问。PDF 指定了一个标准的基于密码的安全处理器，所有的消费者应用程序都需要支持，但应用程序也可以提供它们自己的安全处理器.

SubFilter 指定加密字典内容的语法。它允许处理程序之间的互操作性，也就是说，如果处理器都支持 **SubFilter** 指定的格式，那么文档可以被除了优选的处理器（**Filter** 条目）之外的处理器解密。

V 条目，在指定要使用的加密算法时，需要确定加密密钥的长度，PDF 文件中数据的加密(和解密)基于加密密钥.对于 V 值 2 和 3,Length 条目指定加密密钥的确切长度。在 PDF1.5 中，对于 V 值 4，允许安全处理器使用其自己的加密和解密算法，并指定特定流上使用的 crypt 过滤器（见 3.5.4 节，“Crypt 过滤器”）。

3.5.1 一般加密算法

3.5.2 标准安全处理器

标准加密字典

加密密钥算法

密码算法

3.5。3 公钥安全处理器

公钥加密字典

公钥加密算法

3。5.4 Crypt 过滤器

PDF 1。5 引入了 Crypt 过滤器，可以在 PDF 文件中提供更精细的加密控制。使用隐 Crypt 过滤器涉及以下结构：

3。6 文档结构

PDF 文档可以被视为包含 PDF 文件正文部分的层次对象结构。层次结构的根对象是文档的目录字典(见 3。6。1 节“文档目录"）。层次结构中的大多数对象都是字典.例如,文档的每个页面都由页面对象(包括对页面的内容和其他属性的引用）的字典表示,例如其缩略图(8。2。3 节“缩略图"）和所有注释（8.4 节 ,“注释”）。单个页面对象被绑定在称为页面树的结构中（在 3。6。2 节“页面树”中描述），后者又由文档目录中的间接引用指定.层级中的父，子和兄弟关系由字典条目定义，其值是对其他字典的间接引用。图 3.5 说明了对象层次结构.

注意：本节中描述的数据结构，特别是目录和页面字典，将描述文档结构的条目与处理文档

和页面的详细语义的条目相结合。所有在此节列出的条目，其详细描述都在后续的章节中。

3.6。1 文档目录

文档对象层次结构的根是目录字典，其通过 PDF 文件尾部中的 **Root** 条目定位（见 3。4.4 节“文件尾部”）.目录包含对定义文档内容，大纲，文章线索（PDF 1.1），命名目的地和其他属性的其他对象的引用。此外，它还包含有关如何在屏幕上显示文档的信息，例如是否应自动显示其轮廓和缩略图页面图像以及打开文档时是否应显示除第一页之外的某个位置。 表 3.25 显示了目录字典中的条目。

3。6。2 页树

文档的页面通过页树结构进行访问，页树定义了文档中页面的排序。树结构允许 PDF 使用者应用程序仅使用有限的内存来快速打开包含数千页的文档。该树包含两种类型的节点——中间节点和叶子节点，中间节点被称为页面树节点，叶子节点被称为页面对象--其形式在下面的章节中描述.应用程序应该准备好处理由这些节点构建的任何形式的树结构。最简单的结构将由单个页面树节点组成，该节点直接引用文档的所有页面对象。但是，为了优化应用程序性能，Acrobat Distiller 程序构造了一种特殊形式的树，称为平衡树.关于这种形式的树的更多信息可以在数据结构和算法中找到（见参考书目）。

页树节点

表 3。26 显示了一个页树节点所必需的条目。

表 3.26 页树节点的必需条目

键值	类型	描述
Type	名字	（必需）此字典描述的 PDF 对象的类型；必须是 **Pages**。
Parent	字典	（根节点之外必需，必需是间接引用）父页树节点.

Kids	数组	（必需）子页面树节点的引用，数组形式。子页面树节点可以是页面对象或其他的页面树节点。
Count	整型	（必需)子叶子节点（页面对象）的数量。

页对象

页属性继承

3.6。3 名字字典

3。7 内容流和资源

3。7.1 内容流

3。7。2 资源字典

3.8 常用数据结构

3。8。1 字符串类型

文本字符串类型

PDFDocEncoded 字符串类型

字节字符串类型

3。8.2 文本流

3.8.3 日期

3.8.4 矩形

3.8。5 名字树

3。8。6 数字树

3。9 函数

3。9。1 类型 0（采样)函数

3.9.2 类型 2（指数插值）函数

3.9.3 类型 3（连接）函数

3。9.4 类型 4（PostScript 计算器)函数

类型 4 函数中的错误定义

3.10 文件规范

3.10。1 文件规范字符串

绝对和相对的文件规范

转换为平台相关的文件名

文件规范中的多字节字符串

3。10。2 文件规范字典

3。10.3 嵌入式文件流

相关文件数组

3。10。4 URL 规范

3。10.5 集合项

从 PDF 1.7 开始，集合项字典包含集合中特定文件的集合概要字典描述的数据(见 8。2.4 节“集合”）。表 3。45 介绍了集合项目字典中的条目。

3.10。6 维护文件规范

第 4 章 图形

4.1 图形对象

4。2 坐标系

4.2.1 坐标空间

设备空间

用户空间

其他坐标空间

坐标空间之间的关系

4.2。2 通用转换

4.2。3 转换矩阵

4。3 图形状态

4.3。1 图形状态堆栈

4。3。2 图形状态参数的详细信息

线宽 (Line Width)

线端口样式 (Line Cap Style)

线连接样式 (Line Join Style)

斜接限制(Miter Limit)

当两个线段以锐角相遇并且已经将斜接连接指定为线连接样式时,斜角度可能远远超过行进路径的线的厚度.斜角限制对斜角长度与线宽的比率施加最大值（参见图 4.7）.超过限制时,连接将从斜角转换为斜角。

虚线样式 (Line Dash Pattern)

4.3。3 图形状态运算符

4。3.4 图形状态参数字典

4.4 路径构造与绘制

4.4。1 路径构造运算符

三次贝塞尔曲线

4.4.2 路径绘制运算符

描边(Stroking)

填充(Filling)

非零绕数规则

奇偶规则

4.4.3 路径裁剪运算符

4.5 颜色空间

4.5。1 颜色值

4.5.2 颜色空间族

4。5.3 设备颜色空间

DeviceGray 颜色空间

DeviceRGB 颜色空间

DeviceCMYK 颜色空间

4。5.4 CIE—Based 颜色空间

CalGray 颜色空间

CalRGB 颜色空间

Lab 颜色空间

ICCBased 颜色空间

默认颜色空间

CIE—Based 颜色空间的隐式转换

渲染意图

4.5。5 特殊颜色空间

底纹颜色空间

索引颜色空间

分离颜色空间

DeviceN 颜色空间

多色调示例

4。5。6 套印控制

4。5.7 颜色运算符

4。6 底纹

4.6.1 底纹的一般属性

4。6。2 平铺底纹

彩色平铺底纹

原色平铺底纹

4.6.3 渐变底纹

渐变操作符

渐变字典

颜色空间:特别注意事项

渐变类型

类型 1（功能型）渐变

类型 2（轴向）渐变

类型 3（径向）渐变

类型 4 渐变（自由式三角网格高洛德渐变）

类型 5 渐变（点阵式三角网格高洛德渐变）

类型 6 渐变(孔斯曲面网格)

类型 7 渐变(张量积曲面网格)

4.7 外部对象

4。7。1 PostScript XObjects

4。8 图像

4。8.1 图像参数

4。8.2 样本表示

4。8.3 图像坐标系

4.8.4 图像字典

解码数组

图像插值

备用图像

4.8.5 图像掩码

模板掩码

明确掩码

颜色键掩码

4.8.6 内联图像

4。9 XObjects 格式

4。9.1 字典格式

4。9.2 XObjects 组

4。9.3 引用 XObjects

引用 Xobjects 的打印

特别注意事项

4.10 可选内容

4。10.1 可选内容组

可选内容成员关系字典

意图

4.10。2 制作可选图形内容

内容流中的可选内容

XObjects 和注释中的可选内容

4。10.3 配置可选内容

可选内容属性字典

可选内容配置字典

使用和使用应用程序字典

确定可选内容组的状态

第 5 章　文本

5。1 字体的组织和使用

5.1。1 文本显示的基础

5.1.2 实现特殊的图形效果

5.1。3 字形定位和度量

5.2 文本状态参数和操作符

5.2。1 字符间距

5.2.2 字间距

5.2。3 水平缩放

5.2。4 行距

主要参数 T_l 是在未缩放的文本空间单元中测量的.它指定相邻文本行的基线之间的垂直距离，如图 5。9 所示。

5。2。5 文本渲染模式

5.2.6 文本纵向偏移

5。2。7 文本敲除

文本敲除参数 T_k (PDF 1.4)是一个布尔标志,它决定了在透明的成像模型中,文本元素被认为是基本的对象。与其他文本状态参数不同，没有特定的操作符来设置此参数;它只能通过使用 gs 操作符在图形状态参数词典中通过 TK 条目设置(参见 4.3.4 节,“图形状态参数字典”).

5。3 文本对象

5.3.1 文本定位运算符

5.3。2 文本显示运算符

5。3.3 文本空间细节

5。4 字体数据结构介绍

5.5 简单字体

5。5.1 类型 1 字体

标准类型 1 字体(标准 14 字体)

多主体字体

5。5.2 TrueType 字体

5.5.3 字体子集

5。5。4 类型 3 字体

类型 3 字体示例

5.5。5 字符编码

类型 1 字体的编码

类型 3 字体的编码

TrueType 字体的编码

5。6 复杂字体

5.6.1 CID-Keyed 字体概述

5。6.2 CIDSystemInfo 字典

5.6。3 CID 字体

CID 字体的字形选择

CID 字体的字形指标

5.6.4 CMaps(字符映射)

预定义的 CMaps

嵌入式 CMap 文件

CMap 示例和运算符摘要

5。6。5 类型 0 字体字典

CMap 映射

未定义字符处理

5.7 字体描述符

5。7.1 字体描述符标志

6。4.1 半色调屏幕

6。4.2 点函数

6.4.3 阈值数组

6.4。4 半色调字典

半色调类型 1

半色调类型 6

半色调类型 10

半色调类型 16

半色调类型 5

6。5 扫描转换细节

6.5.1 平面度公差

6。5。2 平滑度公差

6。5.3 扫描转换规则

6.5.4 自动行程调整

第 7 章　透明度

7。1 透明度概述

7.2 基本合成计算

7.2。1 合成计算的基本符号

7。2。2 基本合成公式

7.2。3 混合颜色空间

7.2。4 混合模式

7。2.5 α解释

7。2.6 形状和不透明度计算

源形状和不透明度

结果形状和不透明度

7.2.7 基本合成计算摘要

7.3 透明度组

7。3.1 组合成计算符号

7.3。2 组结构和命名法

7.3.3 组合成计算

7。3。4 隔离组

7.3。5 敲除组

7.3.6 页面组

7。3。7 组合成计算摘要

7.4 软掩码

7。4。1 从α组派生软掩码

7.4.2 从亮度组派生软掩码

7.5 指定 PDF 中的透明度

7。5。1 指定源和背景色

7.5.2 指定混合颜色空间和混合模式

7.5.3 指定形状和不透明度

对象形状和不透明度

掩码形状和不透明度

常量形状和不透明度

7。5.4 指定软掩码

软掩码字典

软掩码图像

7。5。5 透明度组和 XObjects

7。5.6 底纹和透明度

7.6 颜色空间和渲染问题

7。6.1 透明度组的颜色空间

7。6.2 专色和透明度

7.6.3 套印和透明度

混合模式和套印

与不透明套印兼容

特殊路径绘制注意事项

套印行为摘要

7.6。4 渲染参数和透明度

半色调和传递函数

渲染意图和颜色转换

7.6.5 PostScript 兼容性

第 8 章　交互功能

8。1 查看器首选项

8.2 文档级导航

8.2.1 目的地

显式目的地

命名目的地

8.2.2 文档大纲

8。2。3 缩略图

8。2。4 集合

8.3 页面级导航

8。3.1 页面标签

8。3。2 文章

某些文档可能包含逻辑上相关但物理上不连续的内容序列。例如，一个新闻故事，可以从一个时事通讯的第一页开始，并且可以跳转到一个或多个内部的非连续页面上。为了表示这种物理上不连续但逻辑上相关的条目的序列，PDF 文档定义了文章（PDF1。1），一个 PDF 文档中可能包含一个或多个文章。文章内容的顺序由文章线定义，构成文章的各个内容项目被称为线珠。PDF 阅读器应可以提供导航功能,以允许用户按照文章线从一个线珠找到下一个线珠。

8。3。3 演示

子页面导航

8.4 注释

注释将诸如注解,音频或视频等对象与 PDF 文档页面上的位置相关联，或提供通过鼠标和键

盘与用户交互的方式.PDF 包含各种各样的标准注释类型,详见第 8。4.5 节“注释类型"。

许多标准注释类型可以显示在打开状态或关闭状态。关闭时，根据具体的注释类型，它们会以某种独特形式出现在页面上，例如图标,框或橡皮图章. 当用户通过单击来激活注释时，它会显示其关联的对象，例如打开一个弹出窗口，显示文本注释（图 8。2）或者播放音频或视频。

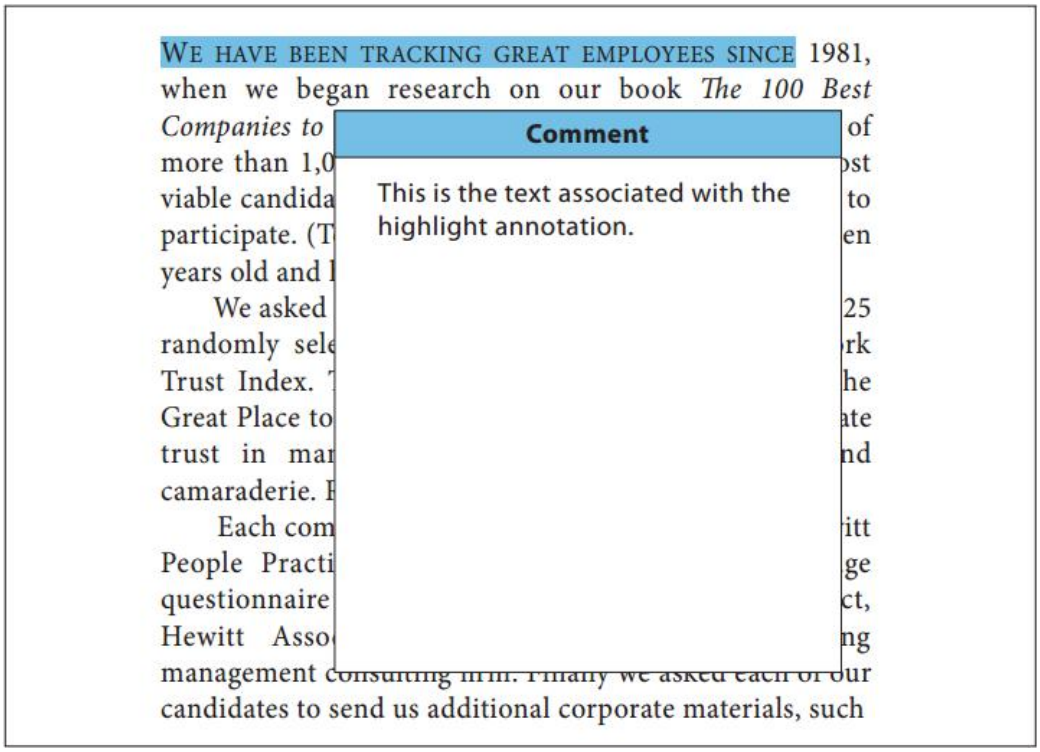

图 8.2 打开注释

查看器应用程序可以允许用户通过使用键盘（特别是 tab 键）在页面上浏览注释；见附录 H 中的实施说明 77。从 PDF1.5 开始，PDF 制作者可以使用页面对象中可选的 Tabs 条目，给导航明确排序(见表 3.27）。以下是此条目的可能值：

- R（行顺序）:注释以行的方式访问.行中的方向由查看器首选项字典中的 Direction 条目确定（参见 8.1 节，“查看器首选项”）。访问的第一个注释是最上面一行的第一个注释。当遇到一行的末尾时，下一行中的第一个注释被访问.
- C（列顺序）：注释以列的方式访问。列由查看器首选项字典中的 Direction 条目排序(参见 8。1 节,“查看器首选项”）。访问的第一个注释是第一列顶部的注释。当遇到列的末尾时，将访问下一列中的第一个注释。

- S（结构顺序）：注释以结构树中出现的顺序访问(参见第 10。6 节“逻辑结构”）.未包含在结构树中注释的顺序与应用程序有关。

注意：上面的描述，都假设页面是以 Rotate 条目指定的方向查看。

每个注释类型的行为由称为注释处理程序的软件模块实现。标准注释类型的处理程序直接构建到 PDF 查看器应用程序中；附加类型的处理程序可以作为插件扩展提供。

8.4。1 注释字典

8.4.2 注释标志

8.4.3 边框样式

8.4.4 外观流

8.4。5 注释类型

标记注释

注释状态

文本注释

链接注释

自由式文本注释

线形注释

矩形和椭圆注释

多边形和折线注释

文本标记注释

插入符号注释

可删除印章注释

毛笔注释

弹出式注释

文件附件注释

音频注释

视频注释

屏幕注释

构件注释

打印标记注释

捕获网络注释

捕获网络注释（PDF 1.3)定义了 PDF 文档页面的捕获特征.（捕获是沿着色彩边界向页面添加标记的过程，以避免在打印页面时由于着色剂重新排列而引起的不必要的视觉伪像。）页面最多可能有一个捕获网络注释，其子类型条目具有值 TrapNet 和它始终是页面对象的 Annots 数组中的最后一个元素（见 3。6.2 节中的“页面对象”）。见 10.10。5 节“捕获支持”，会进一步讨论。

水印注释

8.5 操作

8。5。1 操作字典

8。5。2 触发事件

8。5.3 操作类型

跳转操作

远程跳转操作

嵌入式跳转操作

启动操作

文章线操作

URI 操作

音频操作

视频操作

隐藏操作

命名操作

设置 OCG 状态操作

呈现操作

转换操作

跳转 3D 视图操作

8.6 交互式表单

8。6.1 交互式表单字典

8。6.2 域字典

域名

可变文本

富文本字符串

8。6.3 域类型

按钮域

按钮

复选框

单选按钮

文本域

选择域

签名域

签名域（PDF 1.3）是包含数字签名的表单域（参见第 8。7 节“数字签名”）。表示签名域的域字典可以包含表 8。81 中列出的附加条目以及表 8.69 中描述的标准条目。字段类型(FT)是 Sig，字段值(V)是包含签名并指定签名域各种属性的签名字典（参见表 8.102）。

填写（签名)签名域需要至少更新 V 条目以及通常还关联相关组件注释的 AP 条目。导出签名字段通常会导出 T，V 和 AP 条目.

与其他字段一样，一个签名域实际上可以由一个构件注释字典来描述，它包含与注释和域相关的条目(参见 8。4.5 节中的“构件注释”）。这样字典中的矩形(Rect）注释可以给出域在其页面上的位置。不可见的签名域应该具有零高度和零宽度的矩形注释.

签名域的构件注释的外观字典(AP）定义了该字段在页面上的可视外观（参见 8。4。4 节“外观流"）。有关 Acrobat 如何处理数字签名外观的信息，请参见技术说明数字签名外观（参见参考书目）。

表 8.8。1 签名域指定的附加条目

键值	**类型**	**说明**
Lock	字典	（可选;必须是间接引用;PDF — 1.5）一个签名域锁字典，它指定在签名域签名时锁定的一组表单字段。表 8。82 列出此字典中的条目。
SV	字典	（可选；必须是间接引用;PDF — 1.5）种子值字典 （见表 8.83），其中包含限制应用于此字段的签名属性的信息。

域字典中 SV 条目的值是种子值字典，种子值字典中的条目（见表 8。83)提供了在应用签名时要使用的约束信息。其 Ff 条目指定字典中的其他条目是否必需使用,或者仅仅建议使用。

注意：种子值字典可以包括属于多个处理程序的私有条目的种子值。给定的处理程序应该仅使用与自身相关的那些条目，并忽略其他条目。

表 8。82 签名域锁字典条目

键值	类型	说明
Type	名字	（可选）此字典描述的 PDF 对象的类型； 如果存在,签名域锁字典必须是 SigFieldLock。
Action	名字	（需要）一个名称,与 Fields 一起,指示应该锁定的字段集。有效值: All 文档中所有的域 Include 所有在 Fields 中指定的域 Exclude 所有在 Fields 中没有指定的域
Fields	数组	（当 Action 的值为 Include 或 Exclude 时，则必需）一个包含域名称的文本字符串数组.

表 8.83 签名域种子值字典条目

键值	类型	说明
Type	名字	（可选）此字典描述的 PDF 对象类型； 如果存在，则种子值字典必须是 SV。
Filter	名字	（可选）用于在签名域上签名的签名处理程序。从 PDF 1。7 开始，如果指定了 Filter，Ff 条目指定了此条目是必需的约束条件，则签名时必须使用该条目指定的签名处理程序；否则，不得签名。如果 Ff 条目指定这是一个可选的约束条件,这个处理程序应该在可用的情况下被使用。 如果不可用，则可以使用不同的处理程序.
SubFilter	数组	（可选）名称数组,指定了当签名时使用的编码。数组中与签名处理程序支持的编码相匹配的名字应该

		是实际用于签名的编码。如果指定了 SubFilter,并且 Ff 条目指定了此条目是必需的约束,则在签名时必须使用第一个匹配编码；否则，不得签名。如果 Ff 指定了这是一个可选的约束,那么如果可用的话,则应该使用第一个匹配编码。如果不可用，可以使用不同的编码.
DigestMethod	数组	（可选； PDF 1.7）名称数组，指定了在签名时可以使用的杂凑算法。有效值为 SHA1，SHA256，SHA384,SHA512 和 RIPEMD160。默认值是具体实现的。 *注意:此属性仅适用于数字证书签名包含 RSA 公钥/私钥.如果它包含 DSA 公共/私钥，那么杂凑算法总是 SHA1,这个属性被忽略。*
V	实数	（可选）签名域种子值字典解析器的最低要求能力。值为 1 指定解析器必须能够识别 PDF 1.5 中指定的所有种子值字典条目。值 2 指定它必须能够识别 PDF 1。7 及更早版本中指定的所有种子值字典条目。 Ff 条目指示这是否是必需的约束。 *注意:PDF 规范第五版（PDF 1。6)及更早版本，错误地规定 V 条目为整数类型。这个条目是实数类型。*
Cert	字典	（可选）证书种子值字典(见表 8。84)，其中包含有关签名时要使用的证书的信息。
Reasons	数组	（可选)文本字符串数组,指定了签名文档中的可能原因。如果指定,则此条目中提供的原因将替换查看

器应用程序中使用的原因.Ff 条目指定是否必须在签名中使用数组中的一个原因。

- 如果提供了 Reasons 数组,Ff 条目指定了 Reasons 是必需的约束，数组中的一个原因必须用于签名字典；否则，不得签名。如果 Ff 条目指定了 Reasons 是可选的约束,则可以选择数组中的一个原因，或者可以提供自定义原因。
- 如果忽略了 Reasons 数组或包含一个 0 字符长度的字符串，并且 Ff 条目指定了 Reasons 是必需的约束条件，则必须从签名字典中省略 Reason 条目(参见表 8.102）.

MDP 字典 （可选；PDF 1。6）一个包含单个条目的字典，其键值为 P，其值为 0 到 3 之间的整数。值为 0 将签名定义为普通(非作者）签名（请参见第 8。7 节“数字签名”).值 1 到 3 用于作者签名,并对应于 DocMDP 转换参数字典中 P 的值（见表 8.104）。

如果此条目不存在或不包含 P 条目，则对签名类型或其权限，不会定义任何规则.

TimeStamp 字典 （可选；PDF 1.6）时间戳字典,包含两个条目：

URL 指定时间戳服务器 URL 的 ASCII 码字符串，提供符合 RFC 3161，Internet X。509 公钥基础设施时间戳协议（参见参考书目）的时间戳。

Ff 整数,值为 1(签名需要有时间戳）或 0（签名不需要有时间戳)。默认值：0。

LegalAttestation	数组	（可选； PDF 1.6）指定可能的合法证明的文本字符串数组（参见第 8.7.4 节“法律内容证明”）。 Ff 条目中相应标志的值表示这是否为必需的约束。
AddRevInfo	布尔	（可选; PDF 1。7）是否应执行撤销检查的标志。 如果 AddRevInfo 为 true,则在签名签名域时，查看器应用程序将执行以下附加任务: ● 执行用于签名的证书（和相应的颁发证书）的吊销检查。 ● 在签名值中包含吊销信息。 如果 SubFilter 是 adbe.pkcs7.detached 或 adbe.pkcs7。sha1，则值为 true 时才有意义。如果 SubFilter 是 x509.rsa_sha1,则该条目必须被省略或设置为 false; 否则,签名过程可能会失败. 如果 AddRevInfo 为 true，并且 Ff 条目指定了这是必需的约束,则必须执行上述任务。如果无法执行，则签名失败。 默认值:false。
Ff	整形	（可选)位标志，指定此字典中特定条目的解释。标志值为 1 表示相关联的条目是必需的约束。值为 0 表示相关联的条目是可选约束.位位置:1(Filter)；2(SubFilter）；3（V）； 4（Reasons)；5(LegalAttestation）； 6（ AddRevInfo)；7(DigestMethod）。默认值：0。

表 8。8.4 证书种子值字典条目

键值	类型	说明
Type	字典	(可选)此字典描述的 PDF 对象类型； 如果存在，则证书种子值字典必须为 SVCert。
Subject	数组	(可选)字节字符串数组,包含以 DER 编码的可签名的 X。509v3 证书。X。509v3 证书在 RFC 3280，Internet X。509 公共密钥基础设施，证书和证书吊销列表（CRL)配置文件中进行了描述（参见参考书目）。Ff 条目中相应标志的值表示这是否为必需的约束。
SubjectDN	字 典 数组	（可选；PDF 1。7）一个字典数组，其中每个字典包含键值对,指定证书中必须存在的主题可分辨名称（DN），以使其可签名。证书必须至少包含字典中指定的所有属性。也就是说，证书可以包含其他属性。主题可分辨名称在 RFC 3280 中有描述(参见参考书目）.密钥可以是任何合法的属性标识符.属性名称通常为“cn”，“o”，“email"，“2.5.4。43"的形式，并且始终包含 a—z,A—Z，0-9 和'.’中的字符.值为文本字符串。 一个示例字典是<〈 ／cn（John Smith）/1。5。4.43（JS）〉〉.Ff 条目中相应标志的值表示此条目是否为必需约束。
KeyUsage	ASCII 字 符 串 数 组	（可选； PDF 1.7)ASCII 字符串数组，其中每个字符串指定签名证书中必须存在的可接受的密钥使用扩展名。多个字符串指定一系列可接受的密钥使用扩展。关键字扩展在 RFC 3280 中描述（参见参考书目）. 字符串中的每个字符表示密钥使用类型，其中字符的顺

序表示其表示的密钥使用扩展。数组中的第一到第九个字符从左到右表示以下密钥使用扩展的必需值:

1 数字签名　　4 数据加密　　7 cRL 签名

2 不可否认性　5 密钥协议　　8 仅加密

3 密钥加密　　6 密钥证书　　9 仅解密

任何其他字符都将被忽略。任何缺失的字符或不是以下值的字符都应设置为“X”.支持的字符值:

0　不能设置相应的密钥使用类型。

1　必需设置相应的密钥使用类型.

X　相应的密钥使用类型状态不重要.

例如，字符串值“1”和“1XXXXXXXX”表示必须设置密钥使用类型数字签名并且所有其他密钥使用类型的状态不重要。

Ff 条目中相应标志的值表示这是否为必需的约束。

Issuer　数组　（可选）字节字符串数组，包含以 DER 编码的可接受发行的 X.509v3 证书.如果签名者的证书链接到任何指定的发行人（直接或间接），则该证书被认为是可以签名的。Ff 条目中相应标志的值表示这是否为必需的约束。

OID　数组　（可选）字节字符串数组，包含签名证书中必须存在的证书策略对象标识符(OIDs）。这样一个字符串的例子是（2。16。840。1.113733.1.7.1。1）.仅当 Issuer 的值不为空时，此字段才可用。证书策略扩展在 RFC 3280 中描述(参见参考书目）。Ff 条目中相应标志的值表示这是否为必需的约束。

URL	ASCII 字符串	（可选）一个 URL,其用途由 URLType 条目定义.
URLType	名字	（可选； PDF 1。7）URL 条目用法的名称。URL 有标准用法，也可以实现特定用法。以下值指定有效的标准用法: Browser — 如果没有找到匹配的凭证，URL 引用应该显示在 web 浏览器中的内容，以允许注册一个新的凭据. 这个用法忽略了 Ff 属性的 URL 位。 下面的值指定了一个有效的实现特定用法,定义为 Adobe 系统使用: ASSP - URL 引用可用于基于服务器签名的 Web 签名服务。如果 Ff 属性的 URL 位表示这是必需的约束，这意味着签名时使用的凭据必须来自此服务器。 第三方可以使用自己的属性值来扩展此属性的使用，这些属性值必须符合附录 E 中所述的准则. 默认值是 Browser。
Ff	整形	（可选）一组位标志，指定此字典中特定条目的解释。 标志值为 1 表示签名者仅需要使用条目的指定值。值为 0 表示允许其他值。位位置: 1 (Subject)； 2 (Issuer)； 3(OID)； 4(SubjectDN)； 5 (保留)； 6 (KeyUsage)； 7(URL)． 默认值 0。

8.6。4 表单操作

交互式表单除了第 8。5。3 节“操作类型"中所述之外,还支持四种特殊类型的操作:

- 提交表单操作将所选交互式表单域的名称和值传输到指定的统一资源定位符(URL),可以是要处理它们并发回响应的 Web 服务器的地址.
- 重置表单操作将所选交互式表单域重置为其默认值。
- 导入数据操作将表单数据格式（FDF）数据从指定的文件导入到文档的交互式表单中.

提交表单操作

提交表单操作将所选交互式表单字段的名称和值传输到指定的统一资源定位符(URL），可能是要处理它们并发回响应的 Web 服务器的地址。表 8。85 显示了特定于此类操作的操作字典条目.

操作字典的 Flags 条目的值是一个无符号的 32 位整数,其中包含指定操作的各种特征的标志。标志中的位位置从 1（低位）到 32(高位）编号。表 8.86 显示了标志位的含义；所有未定义的标志位都被保留，并且必须设置为 0。

重置表单操作

导入数据操作

JavaScript 操作

8.6.5 命名页面

8。6。6 表单数据格式化

FDF 文件结构

FDF 文件头部

FDF 文件正文

FDF 文件尾部

FDF 文件目录

FDF 文件域

FDF 文件页

FDF 文件注释字典

8。6.7 XFA 表单

8.7 数字签名

数字签名（PDF 1。3)可用于验证用户的身份和文档的内容。它在签名时存储有关签名者的信息和文档的状态。签名可以是纯数学的，例如公钥/私钥加密的文档摘要，或者它可以是身份

识别的生物特征形式，例如手写签名,指纹或视网膜扫描。所使用的特定身份验证形式由插件签名处理程序实现。鼓励第三方处理程序作者使用 Adobe 注册处理程序名称； 见附录 E。

签名信息包含在签名字典中，其条目列于表 8.102 中。签名处理程序可以使用或省略在表中标记为可选的条目,但如果使用它们，则推荐以标准方式使用它们。此外,签名处理程序可能会添加自己的私有条目.为了避免名称重复，建议所有这些私人条目的密钥前缀为注册的处理程序名称，后跟一个句号（.）。

签名是通过计算文档中的数据摘要（或数据的一部分）来创建的，并将摘要存储在文档中。为了验证签名，将对摘要进行重新计算,并与文档中存储的内容进行比较。摘要值的差异表明，文档自签名以来已经进行了修改。

有两种定义的技术，用于计算 PDF 文件的全部或部分内容的可重复摘要：

- 在文件中的字节范围内计算字节范围摘要,由签名字典中的 ByteRange 条目指定。该范围通常是整个文件，包括签名字典，但不包括签名值本身（Contents 条目）。当存在字节范围摘要时，签名字典中的所有值都必须是直接对象。 见附录 H。的执行说明 139。
- 对象摘要（PDF 1。5)是通过选择性地在内存中移动对象子树来计算的，从引用对象开始，该对象通常是根对象。得到的摘要以及关于如何计算它的信息被放置在一个签名引用字典中，其条目列在表 8。103 中。TransformMethod 条目指定用于计算摘要的通用方法，TransformParams 条目指定计算的可变部分。转换方法在第 8。7。1 节“转换方法”中有详细描述。

PDF 文档可能包含以下标准类型的签名：

- 一个或多个文件(或普通)签名.这些签名显示在签名表单字段中（请参阅 8。6。3 节中的“签名域”）.与每个签名相对应的签名字典是表单字段的值（由其 V 条目指定）.如上所述，签名字典必须包含表示字节范围摘要的 ByteRange 条目。通过重新计算摘要并将其与存储在签名中的摘要进行比较来验证签名.

注意：如果通过增量更新修改并保存签名的文档（请参见第 3。4。5 节“增量更新”）,将

保留与原始签名的字节范围相对应的数据.因此,如果签名有效,则可以在签名时重新创建文档的状态。

- 最多一个 MDP（修改检测和预防）签名（PDF 1.5），也称为作者或认证签名。 MDP 签名的签名字典必须是签名域的值,并且必须包含 ByteRange 条目.也可以从权限字典中的 DocMDP 条目引用（请参见第 8。7。3 节“权限"）.签名字典必须包含具有 DocMDP 变换方法的签名参考字典（见表 8.103)。有关如何创建和验证这些签名的信息，请参见 8.7。1 节中的“DocMDP”。

  用于 MDP 或普通签名的签名字典也可以具有使用 FieldMDP 变换方法的签名参考字典；请参阅 8。7.1 节中的“FieldMDP”。

- 最多两个使用权签名（PDF 1。5）。其签名字典从许可字典中的 UR 或 UR3（PDF 1。6)条目引用（而不是从签名字段）引用； 见表 8.107。字典必须包含具有 UR 变换方法的签名参考字典。有关如何创建和验证这些签名的信息，请参阅 8.7.1 节中的“UR”。

- FDF 文件目录中的 Sig 条目（请参阅 8.6.6 节中的“FDF 目录”)指定签名字典。

表 8。102 签名字典条目

键值	类型	说明
Type	名字	（可选）此字典描述的 PDF 对象的类型；如果存在，签名字典必须是 **Sig**.
Filter	名字	（必需;可继承）验证此签名时要使用的首选签名处理程序的名称.如果 Prop_Build 条目不存在,它也是用于创建签名的签名处理程序的名称。如果 Prop_Build 存在，它可以用于确定创建签名的处理程序的名称（通常与 Filter 相同但不需要）。只要支持指定的 SubFilter 格式,应用程序可以在验证签名时替换不同的处理程序.示例签名处理程序是

		Adobe。PPKLite，Entrust。PPKEF,CICI.SignIt 和 VeriSign。PPKVS。
SubFilter	名字	（可选）描述签名字典中签名值和密钥信息的编码的名称。应用程序可以使用支持此格式的任何处理程序来验证签名. PDF 1。6 定义了公钥加密签名的以下值：adbe。x509.rsa_sha1，adbe.pkcs7。detached 和 adbe。pkcs7.sha1（请参见第 8.7。2 节“签名互操作性”）。其他值可以由第三方开发人员定义，但受限于所有以 **adbe** 开头的名称。前缀将保留给 PDF 的未来版本.所有第三方名称必须在 Adobe Systems 中注册（见附录 E）。
Contents	字节字符串	（必需）签名值。当存在 **ByteRange** 时，该值是十六进制字符串（参见 3。2.3 节中的“十六进制字符串”），表示字节范围摘要的值.如果不存在 **ByteRange**，则该值是签名字典的对象摘要，不包括“**Contents**"条目。 对于公钥签名，内容通常是 DER 编码的 PKCS＃1 二进制数据对象或 DER 编码的 PKCS＃7 二进制数据对象。
Cert	字节字符串数组	（当 **SubFilter** 是 **adbe.x509.rsa_sha1** 时需要）字节字符串数组，表示在签名和验证使用公钥密码技术的签名时使用的 X。509 证书链。如果该链只有一个条目,则是一个字节的字符串.签名证书必须首先

		出现在数组中；它用于验证 **Contents** 中的签名值，其他证书用于验证签名证书的真实性。 如果 SubFilter 是 adbe。pkcs7。detached 或 adbe。pkcs7.sha1，则不会使用此条目,证书链必须放在 Contents 中的 PKCS＃7 中.
ByteRange	数组	（对于作为签名域的一部分的所有签名和从权限字典中的 **UR3** 条目引用的使用权签名，都是必需的）描述摘要计算的确切字节范围的整数对(起始字节偏移,字节长度）数组。多个不连续的字节范围用于描述不包括签名值(**Contents** 条目）本身的摘要。
Reference	数组	（可选; PDF 1.5）签名引用字典数组（见表 8。103）。
Changes	数组	（可选）三个整数的数组，指定在上一个签名和该签名之间进行的对文档的更改:按此顺序,更改的页数，更改的字段数以及填充的字段数（ 见附录 H 中的实施说明 139）。 *注意:签名的顺序由* ***ByteRange*** *的值决定.由于每个签名都会生成增量保存，所以后来的签名具有更大的长度值。*
Name	文 本 字 符 串	（可选）签名文档的人或机构的名称。只有当不可能从签名中提取名称时,才使用该值； 例如，从签字人的证书。
M	日期	（可选）签名时间.根据签名处理程序，这可能是正常的未验证的计算机时间或以可验证的方式从安全时间服务器生成的时间。

		只有当签名时间不可用时，才使用此值。例如,时间戳可以嵌入到 PKCS＃7 二进制数据对象中（请参阅 8。7。2 节中的“PKCS＃7 签名”）.
Location	文本字符串	（可选）签名的 CPU 主机名或实际位置.
Reason	文本字符串	（可选）签名的原因,如(我同意.。。。）.
ContactInfo	文本字符串	（可选）签名人提供的信息，以使收件人能够联系签名者以验证签名；例如，电话号码。
R	整形	（可选）用于创建签名的签名处理程序的版本。 *注意:从 PDF 1。5 开始,此条目已弃用，信息应存储在 **Prop_Build** 字典中。*
V	整形	（可选； PDF 1.5)签名字典格式的版本.它对应于 **SubFilter** 值的上下文中签名字典的使用.如果 **Reference** 字典被认为对签名的验证至关重要，则值为 1。默认 0.
Prop_Build	字典	（可选; PDF 1.5）签名处理程序可以使用的字典来记录捕获用于签名的计算机环境状态的信息，例如用于创建签名的处理程序的名称，软件构建日期，和操作系统. Adobe 发布单独的规范,Acrobat 6.0 的 PDF 签名构

		建字典规范，该规范为该字典的使用提供了实施指南。
Prop_AuthTime	整形	（可选； PDF 1。5）自签署者上次验证后的秒数。它旨在用于签名抵销的索赔.如果值未知，则应省略。
Prop_AuthType	名字	（可选； PDF 1。5）用于验证签名者的方法。它旨在用于签名抵销的索赔.有效值包括 PIN,密码和指纹。

注意：签名字典中的条目可以被概念化为不同的字典；由于历史和加密原因，它们在一个字典中．类别是签名属性(R， M， Name， Reason,Location， Prop_Build， Prop_AuthTime 和 Prop_AuthType)； 关键信息(Cert 和签名值为 PKCS＃7 对象时的内容部分)；引用（Reference 和 ByteRange)；和签名值（Contents，当签名值是 PKCS＃1 对象时).

表 8.103 签名引用字典条目

键值	类型	说明
Type	名字	（可选)此字典描述的 PDF 对象的类型；如果存在,签名引用字典必须是 SigRef。
TansformMethod	名字	（必需）转换方法的名称(参见第 8.7.1 节，“转换方法”）,指导对象在验证签名时所进行的计算或修改分析。有效值： **DocMDP** 用于检测与文档发起者签署的签名字段相关的文档的修改；参见 8.7.1 节中的“DocMDP”。

	UR	用于检测有权限功能的文档的修改，这类文档可使签名无效。参见 8.7.1 节中的“UR"。
	FieldMDP	用于检测对 **TransformParams** 中指定的表单域列表的修改；参见 8.7。1 节中的“FieldMDP"。
	Identity	在签名单个对象时使用，该对象由签名引用字典中的 **Data** 值指定（参见表 8。103）。 此转换方法支持 FDF 文件的签名。有关详细信息，参见 8。7.1 节中的“Identity”.
TansformParams	字典	(可选)用于指定由 TransformMethod 指定的变换方法的转换参数(变量数据)的字典。除 Identity 之外的每个方法都有自己的一组参数.有关各个转换参数字典的详细信息,请参阅上面指定的每个部分。
Data	（各个）	（当 TransformMethod 为 FieldMDP 或 Identity 时需要）对文档中的对象进行间接引用，通过该对象计算摘要或执行对象修改分析。对于 FieldMDP 和 Identity 之外的转换方法,此对象是隐式定义的。
DigestMethod	名字	（可选）标识计算摘要时要使用的算法的名称.有效值为 MD5 和 SHA1。（见附录 H 中的实施说明 144)默认值：MD5。

DigestValue	字符串	（在某些情况下需要)存在时，摘要的计算值。 有关何时需要此条目的详细信息,请参见第 8。7。1 节“转换方法".
DigestLocation	数组	（需要DigestValue时使用,TransformMethod为FieldMDP 或 DocMDP）指定 DigestValue 字符串在 PDF 文件中位置的两个整数数组。整数分别代表起始偏移量和字节长度. 当 DigestValue 直接写入 PDF 文件时,不需要对文档执行任何的加密，则此条目是必需的。指定时，必须使用这些值在验证期间直接从文件读取 DigestValue.

8。7.1 转换方法

转换方法以及转换参数确定在对象摘要计算或修订比较中包含和排除哪些对象。以下部分讨论转换方法的类型，它们的转换参数以及何时被使用.附录 I“对象摘要的计算”详细描述了计算对象摘要的算法。

注意:所有转换方法都从对象摘要中排除签名字典。

DocMDP

DocMDP 转换方法用于检测相对于由文档的作者(应用第一个签名的人)签名的签名字段的修改.文档只能包含一个含有 DocMDP 转换方法的签名域； 它必须是文档中的第一个签名域。它使作者能够指定文档允许做什么更改，什么更改使作者的签名无效。

如前所述，“MDP”代表修改检测和预防.这种签名能够检测作者指定的不允许的修改。此外，

当签名字典被权限字典中的 **DocMDP** 条目引用（参见 8。7.3 节，“权限"）时，也可以防止不允许的更改.

注意：当创建一个作者签名时,鼓励应用程序创建一个合法的认证字典（见第 8.7。4 节“法律内容认证”），它指定了可能导致文档内容出现意外呈现的所有内容，以及作者对这些内容的认证。如果文件的完整性受到质疑,则可以使用该字典来确定作者的意图。

DocMDP 转换参数字典中的 **P** 条目（见表 8。104）表明作者的规范，规定了对文档哪些修改将使签名无效。（如果签名字典是从权限字典中的 DocMDP 条目引用，则也会阻止对文档的这些修改。）P 值为 1 表示文档是最终的;也就是说，任何修改都将使签名无效。值 2 和 3 表示允许对表单域或注释修改。

DocMDP 对象摘要是在文档中的 PDF 对象的子集上计算的.具体来说,该子集包括不被转换参数字典指定的直接或间接修改的对象.附录 I 描述了对象摘要计算。

验证 MDP 签名

要验证 MDP 签名，应用程序首先验证字节范围摘要。接下来，它验证转换参数允许对文档进行的任何修改,方法是使用以下技术之一：

- PDF 1.5 需要签名时对象摘要的计算值存储在签名引用字典中的 DigestValue 条目中（见表 8.103）。因此,应用程序可以在验证时将此条目与其计算的对象摘要值进行比较。如果值不同，则签名无效.
- 在 PDF 1.6 中,DigestValue 条目不是必需的。一旦字节范围摘要被验证，签名字典中 ByteRange 条目指定的文档部分（见表 8.102）对应于签名时文档的状态.因此，应用程序可以比较文档的签名和当前版本，以查看是否对转换参数不允许修改的对象进行了修改。见附录 H。的执行说明 141。

表 8.104 DocMDP 转换参数字典条目

键值	类型	说明
Type	名字	（可选）此字典描述的 PDF 对象的类型； 如果存在，则转换参数字典必须是 TransformParams。
P	数字	（可选）为此文档授予的访问权限。有效值为： 1 不允许对文件进行修改；文件的任何修改都会使签名无效。 2 允许的修改是填写表单，实例化页面模板和签名；其他修改使签名无效. 3 允许的修改与 2 相同，以及注释的创建，删除和修改； 其他修改使签名无效。 默认值：2
V	名字	（可选）DocMDP 转换参数字典的版本。唯一的有效值是 1.2。(请注意，此值是名称对象,而不是数字。)(请参见附录 H 中的实现说明 145）默认值：1.2.

UR

UR 转换方法用于检测文档的修改，此修改会使从权限字典中的 UR 或 UR3 条目引用的使用权签名无效(见 8。7.3 节“权限”)。使用权签名用于启用额外的交互功能，这些功能在特定的查看器应用程序（如 Adobe Reader)中,默认不可用。签名用于验证权限是由真正的授权机构授予的。转换参数字典（见表 8。105)指定了签名有效时启用的附加权限。如果签名无效，因为文档已经以不允许的方式进行了修改，或签名者的身份未被授予扩展权限,则不会授予其他权限.

Adobe Systems 授予权限，例如，使用公钥密码技术在 Adobe Reader 中启用其他功能。它使用证书颁发机构颁发公钥证书来记录与其建立业务关系的创建者.Adobe Reader 验证权限启用签

名是否使用来自 Adobe 授权的证书颁发机构的证书。 其他 PDF 查看器应用程序可以自由使用相同的机制来实现自己的目的。

验证使用权签名取决于签名字典是否从权限字典中的 UR 或 UR3 条目引用（见附录 H 中的实现说明 142）：

- **UR**：在签名时,应用程序通过文档中 PDF 对象的子集计算对象摘要； 也就是说，对象是未被修改的,无论是直接或间接的修改，即使修改是由转换参数字典指定的允许的操作。附录 I 描述了对象摘要计算。该摘要的计算值存储在签名引用字典中的 DigestValue 条目中（见表 8。103)。应用程序可以在验证时将此条目与其计算的对象摘要值进行比较。如果值不同，则签名无效.

  注意：由于算法复杂（因此容易出错），不推荐使用 UR. 应该使用 UR3（见下文)算法。

- **UR3**（PDF 1.6）：签名字典中的 ByteRange 条目（见表 8。102)必需存在。首先，应用程序验证字节范围摘要以确定由 ByteRange 指定的文档的部分是否对应于签名时文档的状态。接下来,应用程序检查文档的当前版本,以查看是否对转换参数不允许修改的对象进行了修改。

表 8。105 UR 转换参数字典条目

键值	类型	说明
Type	名字	（可选)此字典描述的 PDF 对象的类型； 如果存在，则转换参数字典必须是 TransformParams。
Dcoument	数组	（可选)指定文档的其他文档范围使用权限的名称数组。 唯一定义的值是 FullSave，它允许用户将文档与修改的表单和/或注释数据一起保存。（见附录 H 中的实施说明 143）
Msg	文本字符	(可选)可用于指定任意信息的文本字符串，例如为文档添加使用权限的原因。

串

V	名字	（可选）UR 转换参数字典的版本。 唯一有效的值是 2。2. 如果存在未知版本，则不会启用任何权限.（请注意，此值是名称对象，而不是数字。）（请参见附录 H 中的实现说明 145）默认值：2.2。
Annots	数组	（可选）指定文档的附加注释相关使用权的名称数组.PDF 1。5 及更高版本中的有效名称为创建，删除，修改，复制，导入和导出，允许用户在注释上执行命名操作. PDF 1。6 中添加了以下名称。只有当从权限字典的 UR3 条目引用签名字典时，名称才有效（见表 8.107)： Online 允许在线评论；也就是说，能够从服务器上传或下载标记注释. SummaryView 允许显示一个用户界面，该界面总结了文档中的注释（标记注释）。
Form	数组	（可选）一组名称，指定文档的与表单域相关的使用权。 PDF 1。5 中的有效名称是: FillIn 允许用户保存已完成填写表单的文档。 Import 允许用户以 FDF，XFDF 和文本（CSV／TSV）格式导入表单数据文件。 Export 允许用户将表单数据文件导出为 FDF 或 XFDF 格式. SubmitStandalone 允许用户在 Web 浏览器中未打开文

档时提交表单数据。

SpawnTemplate　允许新页面从命名页面模板中实例化。

PDF 1.6 中添加了以下名称。 只有签名字典是从权限字典的 **UR3** 条目引用时才允许使用；请参见表 8.107（参见下面的 FormEx）：

BarcodePlaintext　允许将文本表单域数据编码为纯文本二维条形码。

Online　（PDF 1.6）允许使用特定于表单的在线机制，例如 SOAP 或活动数据对象。

FormEx　数组　（可选;仅当签名字典是从权限字典的 **UR** 条目引用时才允许； PDF 1.5）指定其他表单域的相关使用权限的名称数组. 唯一有效的名称是 BarcodePlaintext，它允许文本表单域数据被编码为纯文本二维条形码。

Signature　数组　（可选）指定文档的附加签名相关的使用权限的名称数组。 唯一定义的值是 Modify，它允许用户将数字签名应用于现有的签名表单域或清除已签名的签名表单域。

EF　数组　（可选； PDF 1.6）一组名称，指定文档中已命名的嵌入文件的附加使用权限。有效的名称是创建，删除，修改和导入，这允许用户对已命名的嵌入文件执行上述名称的操作。

P　布尔　（可选； PDF 1.6）如果为 true，则应将所有消费者应用程序中的文档权限限制为 Adobe Reader 授予的权

限，同时允许此字典中其他条目启用的权限。默认值：false.

FieldMDP

FieldMDP 转换方法通过表单域对象的列表计算对象摘要,并用于检测对这些表单域值的变化。其转换参数字典中的条目列于表 8.106.

表 8.106 FieldMDP 转换参数字典条目

键值	类型	说明
Type	名字	（可选)此字典描述的 PDF 对象的类型； 如果存在，则转换参数字典必须是 **TransformParams**。
Action	名字	(必需)一个名称，与 **Fields** 数组一起,描述哪些表单域包含在对象摘要中（因此在应用签名后不允许修改)。有效值为： All 所有表单域. Include 仅在 **Fields** 中指定的表单域。 Exclude 不在 **Fields** 中指定的表单域。
Fields	数组	(如果 **Action** 是 *Include* 或 *Exclude*，则必需）包含域名称的文本字符串数组。
V	名字	(可选）转换参数字典的版本。PDF 1。5 及更高版本的值为 1。2。 （请注意，此值是名称对象,而不是数字。）（请参见附录 H 中的实现说明 145）默认值：1.2.

在用于表单域工作流的文档中,会发生以下情况：

- 作者可以指定,填写表单域，而不会使作者的签名无效.将 **DocMDP** 转换参数字典的 P 条目值

设置为 2 或 3 时（见表 8.104）.

- 作者还可以指定,在指定的收件人签名文档之后，对指定表单域的任何修改都应使该收件人的签名无效.每个指定的收件人都有一个单独的签名域,每个指定的收件人都有一个相关的签名域锁字典(参见表 8。82），指定该用户锁定的表单域。
- 当收件人签名时，将创建签名,签名引用和转换参数字典。转换参数字典中的“**Action**”和“**Fields**”条目将从签名域锁字典中的相应字段中复制.

  注意:此复制是已完成的，因为签名字典中的所有对象必需是直接对象,如果字典中包含了字节范围签名的话。(即使 ***FieldMDP*** *签名是对象签名，从签名域引用的任何签名字典也必须具有字节范围签名）。因此，转换参数字典不能间接引用签名域锁字典。*

对象摘要是在转换参数字典指定的所有表单域上进行计算，按字母顺序排列（详见附录 I）。指定的表单域被标记为只读来锁定，以防止修改。当验证收件人的签名时,可以检测到表单域的任何更改.

FieldMDP 签名的验证方式与 **DocMDP** 签名类似.有关详细信息,请参阅 8.7。1 节中的“验证 MDP 签名"。

Identity

当计算一个包含所有对象(也就是说,没有对象被排除）的对象摘要时，使用 **Identity** 转换方法。从签名引用字典中的 **Data** 指定的对象(见表 8。103）开始,遍历整个对象树。对象内容的任何修改都会使签名无效。此方法用于支持 FDF 文件的签名。FDF 文件目录是计算摘要的对象。

8.7.2 签名互操作性

PDF 消费者应用程序应当允许签名处理程序之间的互操作性；也就是说，PDF 文件，使用一个供应商的处理程序签名，必须可以使用其他供应商的处理程序进行验证。

签名字典中的 SubFilter 条目指定签名值和密钥信息的编码，而 Filter 条目指定用于验证签名的首选处理程序.处理程序指定它们支持的 SubFilter 编码；因此，除了首选处理程序之外的处理程序,可用于在必要或期望的情况下验证签名。

SubFilter 条目有几个定义的值，全部基于 RSA Security 发布的公钥加密标准，也是互联网工程任务组（IETF）公钥基础设施(PKIX）工作组发布的标准的一部分； 参见参考书目.

PKCS＃1 签名

PKCS#1 标准支持几种公钥加密算法和摘要算法,包括 RSA 加密，DSA 签名以及 SHA—1 和 MD5 摘要(参见参考书目参考)。对于使用 PKCS#1 签名 PDF 文件，SubFilter 唯一推荐的值为 adbe.x509。rsa_sha1,它使用 RSA 加密算法和 SHA—1 摘要算法.签名人的证书链存储在 Cert 条目中.

PKCS＃7 签名

当使用 PKCS#7 签名时，Contents 的值是包含签名的 DER 编码的 PKCS＃7 二进制数据对象.SubFilter 可以使用以下值之一:

- adbe.pkcs7。detached: 没有数据封装在 PKCS＃7 签名数据字段中。
- adbe.pkcs7。sha1:字节范围的 SHA1 摘要封装在具有 ContentInfo 类型为 Data 的 PKCS＃7 签名数据字段中。

PKCS#7 对象必须符合 Internet RFC 2315 PKCS＃7: 加密消息语法，版本 1。5(参见参考书目）中的 PKCS#7 规范。

至少必须包含签名者的 X。509 签名证书。此证书用于验证 Contents 中的签名值。

PKCS＃7 对象可以可选地包含以下属性:

- 时间戳信息为无符号属性（PDF 1.6）:时间标记符号必须符合 RFC 3161,并且必须按照 RFC 3161 的附录 A 中所述计算并嵌入到 PKCS＃7 对象中.

- 吊销信息为签名属性（PDF 1。6)：该属性包括对签名者证书及其颁发者证书执行吊销检查的所需的所有吊销信息。
- 来自签名者信托链的一个或多个颁发者证书（PDF 1。6）；见附录 H.的实施说明 146。
- 与签名者证书相关联的一个或多个 RFC 3281 属性证书（PDF 1.7）。

吊销信息

以下对象标识符标识 Adobe 的吊销信息属性：

adbe-revocationInfoArchival OBJECT IDENTIFIER : :=

{ adbe(1.2.840.113583) acrobat(1) security (1) 8 }

吊销信息属性的值可以包括以下数据类型：

- 证书吊销列表（CRL），见 RFC 3280（参见参考文献）：CRL 通常很大，因此不建议嵌入到 PKCS#7 对象中。
- 在线证书状态协议（OCSP）响应，在 RFC 2560，X.509 Internet 公钥基础设施在线证书状态协议 OCSP（参见参考书目）中描述:这些通常是小型且不变的,并且是建议包括在 PKCS＃7 对象中的数据类型。
- 自定义吊销信息:本规范不规定格式，它将被编码为八进制字符串。应用程序应该能够通过查看相关的对象标识符，来确定八进制字符串中包含的数据类型。

Adobe 的吊销信息属性值具有 ASN。1 类型 RevocationInfoArchival:

```
RevocationInfoArchival :: = SEQUENCE {
crl [0] EXPLICIT SEQUENCE of CRLs, OPTIONAL
ocsp [1] EXPLICIT SEQUENCE of OCSP Responses, OPTIONAL
otherRevInfo [2] EXPLICIT SEQUENCE of OtherRevInfo, OPTIONAL
}
OtherRevInfo ::= SEQUENCE {
```

Type OBJECT IDENTIFIER

Value OCTET STRING

}

对于字节范围签名，**Contents** 是以“〈”和“>”分隔的十六进制字符串。它必须精确地匹配 **ByteRange** 指定的范围之间的空间。由于 PKCS＃7 对象的长度不是全部都可预测的，因此在文件中，将 PKCS＃7 写入分配的空间之前，通常需要在字符串的末尾(“〉”分隔符之前)以零填充 **Contents** 的值。

用于编码签名值的最常用格式是 **adbe。pkcs7.detached**。这个编码允许在算法使用方面有最多的选择。下表显示了各种 **SubFilter** 值支持的算法。

	SubFilter 值		
	adbe.pkcs7 。 detached	**adbe.pkcs7。sha1**	**adbe。x509。rsa。sha1**[a]
消息摘要	SHA1 (PDF 1。3) SHA256 (PDF 1.6) SHA384 (PDF 1。7) SHA512 (PDF 1。7) RIPEMD160 (PDF 1。7)	SHA1 (PDF 1。3)[b]	SHA1 (PDF 1.3) SHA256 (PDF 1。6) SHA384 (PDF 1.7) SHA512 (PDF 1.7) RIPEMD160 (PDF 1.7)
RSA 算法支持	Up to 1024-bit (PDF 1。3) Up to 2048-bit (PDF 1.5) Up to 4096—bit (PDF 1.5)	参见 **adbe 。 pkcs7 。 detached**	参见 **adbe.pkcs7 。 detached**

DSA 算法支持	Up to 4096—bits (PDF 1。6)	参见 adbe 。 pkcs7 。 detached	无

a. 尽管以 SubFilter 值的形式出现 sha1,但支持的编码不限于 SHA1 算法。 PKCS＃1 对象包含一个标识符，指示使用哪种算法.

b. 其他摘要算法可用于对签名数据字段杂凑；然而，SHA1 始终用于对正在签名的数据杂凑.

8。7。3 权限

文档目录（见表 3.25)中的 Perms 条目指定了权限字典（PDF 1。5）。字典中的每个条目（当前已定义的条目,见表 8。107）指定了控制文档访问权限的权限处理器的名称。这些权限类似于由安全性处理程序定义的权限（见 3.5.2 节中的表 3.20）,但不要求对文档进行加密。对于实际授予文档的权限（例如，填写表单域的功能），必须由权限字典中的每个权限处理程序以及安全处理程序允许.

表 8.107 权限字典条目

键值	类型	说明
DocMDP	字典	（可选）对签名字典的间接引用（见表 8.102）。该字典必须包含具有 DocMDP 转换方法的签名引用字典（见表 8。103）的 Reference 条目(见 8.7.1 节中的“DocMDP"）和相应的转换参数。 如果此条目存在,消费者应用程序应执行 DocMDP 转换参数字典中的 P 属性指定的权限，并且还应根据是否违反这些权限来验证相应的签名。
UR	字典	（可选）签名字典，用于指定和验证授予此文档的附加功能(使用权限）；即启用查看器应用程序在默认情况下不

		可用的交互功能。 例如,Adobe Reader 不允许默认保存文档，但 Adobe Systems 可能会授予在 Adobe Reader 中为特定文档保存的权限.该签名用于验证 Adobe Systems 授予的权限。 签名字典必须包含一个 **Reference** 条目,该条目是具有 UR 转换方法的签名引用字典（见 8。7。1 节中的“UR”）.此方法的转换参数字典指示应为文档授予哪些附加权限。如果签名有效，除了应用程序的默认权限之外,Adobe Reader 还允许为文档指定的权限。 此签名字典必需不包含 **ByteRange** 条目。
UR3	字典	（可选; PDF 1.6）指定和验证使用权限的签名字典.除了签名字典必须包含 **ByteRange** 条目外，上面 UR 条目的描述适用于 UR3。有关详细信息,请参阅 8。7。1 节中的“UR”。

8。7。4 法律内容认证

PDF 语言提供了许多功能,可以使 PDF 文档的呈现出现不同。这些功能可能被用来构造一个有意或无意地误导文档接收者的文档。这些情况与在考虑 PDF 文件签名的法律含义时相关。

因此，有必要提供一种机制，让文档接收者可以确定文档是否可信赖。主要方法是仅接受包含作者签名的文档（一个具有 **DocMDP** 签名的文档，该签名可以定义文档中允许修改的内容）；请参见 8。7。1 节中的“DocMDP”）。

创建作者签名时，应用程序还应创建一个合法的认证字典，其条目如表 8。108 所示。该字典是文档目录中 **Legal** 条目的值（见表 3.25）。其条目指定可能导致文档内容意外呈现的所有内容。作者可以通过 **Attestation** 条目进一步澄清此类内容。审查人员应自行确定他们信任文

件的作者和内容.在验证文档的合法性时，可以在本字典信息的上下文中审查任何可疑内容。

表 8.108 法律认证字典条目

键值	类型	说明
JavaScriptActions	整形	（可选)文档中找到的 JavaScript 操作的数量（见 8.6。4 节中的“JavaScript 操作”)。
LaunchActions	整形	（可选）文档中找到的启动操作的数量（见 8.5.3 节中的“启动操作"）。
URIActions	整形	（可选）文档中找到的 URI 操作的数量（见 8.5.3 节中的“URI 操作”）。
MovieActions	整形	（可选）文档中找到的视频操作的数量(见 8。5.3 节中的“视频操作"）。
SoundActions	整形	（可选）文档中找到的音频操作的数量（见 8.5.3 节中的“音频操作")。
HideAnnotationActions	整形	（可选)文档中找到的隐藏操作的数量(见 8。5.3 节中的“隐藏操作”）。
GotoRemoteActions	整形	（可选）文档中找到的远程跳转操作的数量（见 8。5。3 节中的“远程跳转操作”）。
AlternateImages	整形	（可选）文档中找到的备用图像的数量（见 4.8。4 节中的“备用图像"）。
ExternalStreams	整形	(可选）文档中找到的外部流的数量。
TrueTypeFonts	整形	（可选）文档中找到的 TrueType 字体的数量（见 5。5。2 节“TrueType 字体")。
ExternalRefXobjects	整形	（可选）文档中找到的引用 XObjects 的数量（见 4.9.3 节“引用 XObjects”)。

ExternalOPIdicts	整形	（可选）文档中找到的 OPI 字典的数量（见 10.10.6 节“打开印前接口(OPI）"）。
NonEmbeddedFonts	整形	（可选）文档中找到的非嵌入式字体的数量（见 5.8 节“嵌入式字体程序”）。
DevDepGS_OP	整形	(可选)在文档中找到的图形状态参数 **OP** 的引用数(见表 4。8）。
DevDepGS_HT	整形	（可选)在文档中找到的图形状态参数 **HT** 的引用数(见表 4。8）.
DevDepGS_TR	整形	（可选）在文档中找到的图形状态参数 **TR** 的引用数(见表 4.8）。
DevDepGS_UCR	整形	（可选）在文档中找到的图形状态参数 **UCR** 的引用数（见表 4。8）.
DevDepGS_BG	整形	（可选）在文档中找到的图形状态参数 **BG** 的引用数（见表 4.8）。
DevDepGS_FL	整形	（可选）在文档中找到的图形状态参数 **FL** 的引用数（见表 4.8）。
Annotations	整形	（可选)文档中找到的注释的数量（见第 8。4 节“注释”）。
OptionalContent	布尔	（可选)如果在文档中找到可选内容，则为 true（见 4。10 节“可选内容”）.
Attestation	文本字符串	（可选）由文档作者创建的证明，说明本字典中是否存在任何其他条目，或存在影响文档合法完整性的任何其他内容。

8.8 测量属性

8.9 文档需求

8.9.1 需求处理者

第 9 章 多媒体功能

9.1 多媒体

9.1.1 可行性

9.1。2 渲染

媒体渲染

选择器渲染

9。1。3 媒体剪辑对象

媒体剪辑数据

媒体剪辑部分

9。1.4 媒体播放参数

9.1.5 媒体屏幕参数

媒体偏移字典

时间间隔字典

9。1.6 其他多媒体对象

媒体播放器对象

媒体播放器信息字典

软件标记符字典

软件 URIs

版本数组

监视器说明符

9.2 音频

9。3 视频

9。4 交替演示

9。5 3D 功能

9。5.1 3D 注释

9。5.2 3D 流

3D 动画风格字典

3D 引用字典

9。5.2 3D 视图

投影字典

3D 背景字典

3D 渲染模式字典

3D 照明方案字典

3D 截面字典

3D 节点字典

9.5.3 3D 坐标系统

9。5.4 3D 标记

第 10 章 文档交换

本章所描述的功能不影响文档的最终呈现。这些特性使文档能够包含在应用程序之间交换文档时使用的更高级别的信息：

- 程序集(10。1 节）定义了 PDF 操作符的实现.
- 元数据（10.2 节)，包括文档的一般信息或文档的组成部分，例如其标题、作者、创建和修改日期.
- 文件标识符（10。3 节)，用于从一个 PDF 文件到另一个 PDF 文件的可靠引用。
- 页面片字典（10.4 节)允许应用程序将私有数据嵌入到 PDF 文档中以供文档使用。
- 标记内容操作符（10.5 节），用于标识内容流的部分并将其与附加属性或外部指定对象相关联。
- 逻辑结构设施(10.6 节），用于对文档的内容强加一个层次结构的组织。
- PDF 标签(10。7 节），一组使用标记的内容和逻辑结构设施的约定,以便于为其他目的提取和重用文档的内容。

- 增加文件对残疾用户的可访问性的各种方法（10.8 节），包括识别其编写的自然语言（如英文或西班牙文），以利于文本语言引擎.
- Web Capture 插件扩展(10.9 节），它从基于 internet 或本地的 HTML、PDF、GIF、JPEG 和 ASCII 文本文件创建 PDF 文件。
- 支持印前生产工作流程的设施（10.10 节），例如页面边界的规范和打印机标记的生成，色彩分离，输出意图,陷阱和高分辨率图像的低分辨率代理。

10.1 过程集

内容流中使用的 PDF 运算符被归类为称为过程集的相关运算符的类别（见表 10.1）。每个过程集对应于包含该过程集中的运算符的实现的命名资源。内容流资源字典中的 ProcSet 条目（请参见第 3。7.2 节“资源字典”）包含由该内容流中使用的过程集名称组成的数组。只有当内容流打印到 PostScript 输出设备时，才使用这些过程集.这些名称标识必须发送到设备以解释内容流中的 PDF 操作符的 PostScript 过程集。此数组的每个元素都必须是表 10。1 所示的预定义名称之一。（见附录 H 中的实施说明 159）

10.2 元数据

10。2。1 文档信息字典

10.2.2 元数据流

10。3 文件标识符

10。4 页面片字典

10。5 内容标记

10。5.1 属性列表

10.5.2 内容标记和裁剪

10。6 逻辑结构

10.6.1 结构层次

10。6.2 结构类型

10。6.3 结构内容

内容标记序列作为内容项

PDF 对象作为内容项

从内容项中寻找结构元素

10.6。4 结构属性

属性类

属性修订号

用户属性

10。6。5 逻辑结构示例

10。7 标记 PDF

10.7.1 标记 PDF 和页面内容

10.7.2 基本布局模型

10.7。3 标准结构类型

组元素

块级结构元素

段落元素

列表元素

表格元素

块级结构使用指南

内联级结构元素

内联元素

注释元素

Ruby 和 Warichu 元素

插图元素

10.7。4 标准结构属性

标准属性拥有者

属性值和继承

布局属性

一般布局属性

BLSE 布局属性

ILSE 布局属性

内容和分配矩形

插图属性

列属性

列表属性

打印域属性

表格属性

10.8 辅助功能支持

10。8.1 自然语言规范

语言标识符

语言规范层次结构

多语言文本数组

10。8.2 其他说明

10。8.3 替换文本

10.8。4 缩略语和缩略语的扩展

10.9 网络捕获

10。9.1 网络捕获信息字典

10.9.2 内容数据库

URL 字符串

数字标识符

唯一名称生成

10.9.3 内容集

页面集

图像集

10。9.4 源信息

URL 别名字典

命令字典

命令设置

10。9.5 和网络捕获相关的对象属性

10.10 印前支持

10.10。1 页面边界

页面边界显示

10.10.2 打印机商标

10.10.3 分离字典

10.10.4 输出意图

10。10.5 捕获支持

捕获网络注释

捕获网络外观

10。10。6 打开印前接口(OPI)

附录 A 运算符摘要

附录 B 类型 4 函数运算符

B.1 算术运算符

附录 C 实施限制

附录 D 字符集和编码

附录 E PDF 名称注册表

附录 F 线性化 PDF

F.2.2 线性参数字典（Part2）

F.2。3 首页交叉引用表和尾（Part3）

F.2。4 文件目录和文件级对象(Part4）

F.2.5 提示流（Part5 和 10）

F.2。6 第一页（Part6）

F。2.7 剩余页面（Part7）

F。2。8 共享对象（Part8）

F。2.9 其他对象（Part9）

F.2。10 主交叉引用表和尾（Part11）

F。3 提示表

F。3。1 页面偏移提示表

F.3。2 共享对象提示表

F.3.3 缩略图提示表

F。3。4 通用提示表

F.3。5 扩展通用提示表

F。3.6 嵌入式文件流提示表

F。4 访问策略

F。4.1 从第一页打开

F.4.2 从任意页打开

F.4.3 进入开放文档的另一页

F.4。4 逐页绘制页面

F。4。5 跟随文章线

F。4。6 访问更新文件

附录 G PDF 文件示例

G。1 最小的 PDF 文件

示例 G。1 是不绘制任何东西的 PDF 文件； 它几乎是最小的可接受的 PDF 文件。 它不是严格的最小可接受的，因为它包含一个大纲字典（文档目录中的大纲）带有零计数(在这种情况下，该对象通常将被省略）；页面内容流（页面对象中的内容)；以及包含 ProcSet 数组的资源字典（页面对象中的资源）。该文件包含这些对象,成为创建其他更切实际的 PDF 文件的起点。

表 G。1 列出了本例中的对象。

表 G。1 最小示例文件中的对象

对象编号	对象类型
1	Catalog（文档目录）
2	Outlines（大纲字典）

3	**Pages**（页面树节点）
4	**Page**(页面对象)
5	内容流
6	过程集数组

注意:使用示例 G.1 作为创建其他文件的起点时,请记住根据需要更新 ProcSet 数组(参见第 10。1 节“过程集”）。另外，请记住,交叉引用表条目可能需要有一个尾随空格（参见第 3.4。3 节“交叉引用表”）.

示例 G.1

```
%PDF-1。4
1 0 obj
   << /Type /Catalog
     /Outlines 2 0 R
     /Pages 3 0 R
   〉〉
endobj
2 0 obj
    〈< /Type Outlines
      /Count 0
   〉〉
endobj
3 0 obj
   <〈 /Type /Pages
      /Kids [4 0 R]
```

```
      /Count 1
   〉〉
endobj
4 0 obj
   << /Type /Page
      /Parent 3 0 R
      /MediaBox [0 0 612 792]
      /Contents 5 0 R
      /Resources << /ProcSet 6 0 R 〉〉
   >〉
endobj
5 0 obj
   <〈 /Length 35 〉>
stream
…Page—marking operators…
endstream
endobj
6 0 obj
   [/PDF]
endobj
xref
0 7
0000000000 65535 f
0000000009 00000 n
```

```
0000000074 00000 n
0000000120 00000 n
0000000179 00000 n
0000000300 00000 n
0000000384 00000 n
trailer
   << /Size 7
      /Root 1 0 R
   >>
startxref
408
%%EOF
```

G。2 简单文本字符串示例

G。3 简单图形示例

G。4 页面树示例

G。5 大纲层次结构示例

G。6 更新示例

此示例显示被更新了几次的 PDF 文件的结构；它说明了多个文件正文部分,交叉引用部分和尾部。另外,它表明一旦对象被分配了一个对象标识符,它就保留该标识符,直到对象被删除，即使对象被改变了。最后，该示例说明了已删除的对象的交叉引用条目的重用以及对象已删除后生成编号的增加。

原始文件是附录 G.1 中的示例 G.1.更新分为四个阶段，每个阶段后保存文件：

1. 增加 4 个文本注释。
2. 修改其中一个文本注释。
3. 删除其中两个文本注释。
4. 增加三个文本注释.

以下部分显示在每个阶段添加到文件中的段.在整个这个例子中,对象被它们的对象标识符引用，它们由对象编号和生成编号组成，而不是像前面的例子那样简单地通过它们的对象编号.这是必要的，因为示例重用对象编号；因此,它们表示的对象不是唯一的。

注意：这些部分中的表仅显示在更新过程中被修改的对象.在更新过程中未更改的对象未显示。

G.6.1 步骤 1：添加 4 个文本注释

将四个文本注释添加到初始文件中，并保存文件。表 G.4 列出了此更新涉及的对象。

表 G.4 增加 4 个文本注释后的对象使用

对象标识	**对象类型**
4 0	**Page**（页面对象）
7 0	注释数组
8 0	**Annot**(注释字典)
9 0	**Annot**（注释字典）
10 0	**Annot**（注释字典）
11 0	**Annot**（注释字典）

示例 G.7 显示了这次更新增加到文件中的行。页面对象被更新是因为其增加了 **Annots** 条目。注意,文件的尾部现在包含了一个 **Prev** 条目，此条目指向文件中原交叉引用段，而尾部最后的

startxref 值指向了此次更新增加的交叉引用段.

示例 G。7

```
4 0 obj
   〈〈 /Type /Page
    /Parent 3 0 R
    /MediaBox  [0 0 612 792]
    /Contents 5 0 R
    /Resources 〈〈 /ProcSet 6 0 R >>
    /Annots 7 0 R
  >〉
endobj
7 0 obj
   [ 8 0 R
   9 0 R
   10 0 R
   11 0 R
  ]
endobj
8 0 obj
  <〈 /Type /Annot
    /Subtype /Text
    /Rect [44 616 162 735]
    /Contents （ Text ＃1)
    /Open true
```

```
  〉〉
endobj
9 0 obj
  〈〈 /Type /Annot
    /Subtype /Text
    /Rect  [224 668 457 735]
    /Contents  ( Text #2)
    /Open false
  〉〉
endobj
10 0 obj
  << /Type /Annot
    /Subtype /Text
    /Rect [239 393 328 622]
    /Contents  ( Text #3)
    /Open true
  〉〉
Endobj
11 0 obj
  〈〈 /Type /Annot
    /Subtype /Text
    /Rect [34 398 225 575]
    /Contents ( Text #4)
    /Open false
```

```
   〉〉
endobj
xref
0 1
0000000000 65535 f
4 1
0000000632 00000 n
7 5
0000000810 00000 n
0000000883 00000 n
0000001024 00000 n
0000001167 00000 n
0000001309 00000 n
trailer
   〈< /Size 12
    /Root 1 0 R
    /Prev 408
  >〉
startxref
1452
%%EOF
```

G。6.2 步骤 2:修改一个文本注释

G。6。3 步骤 3: 删除两个文本注释

G。6.4 步骤 4:增加三个文本注释

G.7 描述分层列表的结构化元素

G。7。1 内容表

G。7.2 嵌套列表

附录 H 兼容性和实现说明

H。1 PDF 版本号

H.2 功能兼容性

H。3 实现说明

1。2，“PDF 1.7 功能简介”

3.1.2，“注释”

3.2。4，“名字对象"

附录 I 对象杂凑的计算

I。1 基本对象类型

I。2 选择性计算