CHAPTER

Interactive F

This chapter describes the PDF features that allo
ment on the screen, using the mouse and keybo
media features, which are described in Chapter

- *Preference settings* to control the way the doc
  (Section 8.1, "Viewer Preferences")
- *Navigation* facilities for moving through the document in a variety of ways (Sections 8.2, "Document-Level Navigation," and 8.3, "Page-Level Navigation")
- *Annotations* for adding text notes, sounds, movies, and other ancillary information to the document (Section 8.4, "Annotations")
- *Actions* that can be triggered by specified events (Section 8.5, "Actions")
- *Interactive forms* for gathering information from the user (Section 8.6, "Interactive Forms")
- *Digital signatures* that authenticate the identity of a user and the validity of the document's contents (Section 8.7, "Digital Signatures")
- *Measurement properties* that enable the display of real-world units corresponding to objects on a page (Section 8.8, "Measurement Properties")

8.1 Viewer Preferences

TABLE 8.1 Entries in a viewer pref

KEY	TYPE	VALUE
HideToolbar	boolean	*(Optional)* A flag specifyi bars when the document
HideMenubar	boolean	*(Optional)* A flag specif menu bar when the docu
HideWindowUI	boolean	*(Optional)* A flag specify the document's window leaving only the docume
FitWindow	boolean	*(Optional)* A flag specifyi fit the size of the first dis
CenterWindow	boolean	*(Optional)* A flag specifyi in the center of the scree
DisplayDocTitle	boolean	*(Optional; PDF 1.4)* A fl should display the document title taken from the **Title** entry of the document information dictionary (see Section 10.2.1, "Document Information Dictionary"). If **false**, the title bar should instead display the name of the PDF file containing the document. Default value: **false**.
NonFullScreenPageMode	name	*(Optional)* The document's *page mode*, specifying how to display the document on exiting full-screen mode: UseNone — Neither document outline nor thumbnail images visible UseOutlines — Document outline visible UseThumbs — Thumbnail images visible UseOC — Optional content group panel visible This entry is meaningful only if the value of the **PageMode** entry in the catalog dictionary (see Section 3.6.1, "Document Catalog") is FullScreen; it is ignored otherwise. Default value: UseNone.
Direction	name	*(Optional; PDF 1.3)* The predominant reading order for text: L2R — Left to right

KEY	TYPE	VALUE
ViewArea	name	*(Optional; PDF 1.4)* The area of a page to be displ The value is the key desi object (see "Page Objec Boundaries"). If the spec object, its default value i Default value: CropBox. ***Note:*** *This entry is intend interpret or manipulate 10.10.1, "Page Boundarie*
ViewClip	name	*(Optional; PDF 1.4)* The tents of a page are to b screen. The value is the the page object (see "Pa "Page Boundaries"). If th page object, its default v 145. Default value: CropBox. ***Note:*** *This entry is intended primarily for use by prepress applications that interpret or manipulate the page boundaries as described in Section 10.10.1, "Page Boundaries." Most PDF consumer applications disregard it.*
PrintArea	name	*(Optional; PDF 1.4)* The name of the page boundary representing the area of a page to be rendered when printing the document. The value is the key designating the relevant page boundary in the page object (see "Page Objects" on page 144 and Section 10.10.1, "Page Boundaries"). If the specified page boundary is not defined in the page object, its default value is used, as specified in Table 3.27 on page 145. Default value: CropBox. ***Note:*** *This entry is intended primarily for use by prepress applications that interpret or manipulate the page boundaries as described in Section 10.10.1, "Page Boundaries." Most PDF consumer applications disregard it.*

KEY	TYPE	VALUE
PrintClip	name	*(Optional; PDF 1.4)* The tents of a page are to be c is the key designating the "Page Objects" on page the specified page bound value is used, as specifi CropBox. ***Note:*** *This entry is intend interpret or manipulate 10.10.1, "Page Boundarie*
PrintScaling	name	*(Optional; PDF 1.6)* The dialog is displayed for thi cates that the print d AppDefault, which indic print scaling. If this e should use the current pr ***Note:*** *If the print dialog is suppressed and its parameters are provided directly by the application, the value of this entry should still be used.*
Duplex	name	(*Optional; PDF 1.7*) The paper handling option to use when printing the file from the print dialog. The following values are valid: **Simplex** - Print single-sided **DuplexFlipShortEdge** - Duplex and flip on the short edge of the sheet **DuplexFlipLongEdge** - Duplex and flip on the long edge of the sheet Default value: none
PickTrayByPDFSize	boolean	(*Optional; PDF 1.7*) A flag specifying whether the PDF page size is used to select the input paper tray. This setting influences only the preset values used to populate the print dialog presented by a PDF viewer application. If **PickTrayByPDFSize** is true, the check box in the print dialog associated with input paper tray is checked. ***Note:*** *This setting has no effect on Mac OS systems, which do not provide*

KEY	TYPE	VALUE
PrintPageRange	array	(*Optional; PDF 1.7*) The box when the file is print 1. Each pair consists of t number of integers caus cause the entire array to Default value: as defined
NumCopies	integer	(*Optional; PDF 1.7*) The dialog is opened for this f 5. Values outside this ran Default value: as defined

8.2 Document-Level Navigation

The features described in this section allow a P the user with an interactive, global overview of a

- As a hierarchical *outline* showing the document's internal structure
- As a collection of *thumbnail images* representing the pages of the document in miniature form

Each item in the outline or each thumbnail image can be associated with a corresponding *destination* in the document, so that the user can jump directly to the destination by clicking with the mouse.

8.2.1 Destinations

A *destination* defines a particular view of a document, consisting of the following items:

- The page of the document to be displayed

tions" on page 654 and "Remote Go-To Actions
destination specifies the view of the document t
item or annotation is opened or the action is per
OpenAction entry in a document's catalog (Sect
may specify a destination to be displayed when t
nation may be specified either explicitly by an
properties or indirectly by name.

Explicit Destinations

Table 8.2 shows the allowed syntactic forms for s
in a PDF file. In each case, *page* is an indirect ref
dinate values (*left*, *right*, *top*, and *bottom*) are ex
coordinate system. The page's *bounding box* is th
of its contents. (If any side of the bounding box
the corresponding side of the crop box is used i
Boundaries," for further discussion of the crop b

Note: *No page object can be specified for a destination associated with a remote go-to action (see "Remote Go-To Actions" on page 655) because the destination page is in a different PDF document. In this case, the* page *parameter specifies a page number within the remote document instead of a page object in the current document.*

TABLE 8.2 Destination syntax

SYNTAX	MEANING
[*page* /XYZ *left top zoom*]	Display the page designated by *page*, with the coordinates (*left*, *top*) positioned at the upper-left corner of the window and the contents of the page magnified by the factor *zoom*. A null value for any of the parameters *left*, *top*, or *zoom* specifies that the current value of that parameter is to be retained unchanged. A *zoom* value of 0 has the same meaning as a null value.
[*page* /Fit]	Display the page designated by *page*, with its contents magnified just enough to fit the entire page within the window both horizontally and vertically. If the required horizontal and vertical magnification factors are different, use

SYNTAX	MEANING
[*page* /FitH *top*]	Display the page designated tioned at the top edge of the just enough to fit the entire wi for *top* specifies that the curre changed.
[*page* /FitV *left*]	Display the page designated b tioned at the left edge of the just enough to fit the entire h ue for *left* specifies that the c unchanged.
[*page* /FitR *left bottom right top*]	Display the page designated b to fit the rectangle specified entirely within the window b horizontal and vertical magni the two, centering the rectang null value for any of the para
[*page* /FitB]	*(PDF 1.1)* Display the page designated by *page*, with its contents magnified just enough to fit its bounding box entirely within the window both horizontally and vertically. If the required horizontal and vertical magnification factors are different, use the smaller of the two, centering the bounding box within the window in the other dimension.
[*page* /FitBH *top*]	*(PDF 1.1)* Display the page designated by *page*, with the vertical coordinate *top* positioned at the top edge of the window and the contents of the page magnified just enough to fit the entire width of its bounding box within the window. A null value for *top* specifies that the current value of that parameter is to be retained unchanged.
[*page* /FitBV *left*]	*(PDF 1.1)* Display the page designated by *page*, with the horizontal coordinate *left* positioned at the left edge of the window and the contents of the page magnified just enough to fit the entire height of its bounding box within the window. A null value for *left* specifies that the current value of that parameter is to be retained unchanged.

located in another PDF document. For example, ter 6 in another document might refer to the Chap6.begin, instead of by an explicit page num the location of the chapter in the other documen ing the link. If an annotation or outline item th has an associated action, such as a remote go-to tions" on page 655) or a thread action ("Thread nation is in the file specified by the action's **F** e the destination is in the current file.

In PDF 1.1, the correspondence between na defined by the **Dests** entry in the document ca ment Catalog"). The value of this entry is a dicti tination name and the corresponding value i destination, using the syntax shown in Table 8. whose value is such an array. The latter form a associated with the destination, as well as enab Actions" on page 654) to be used as the target of

In PDF 1.2, the correspondence between strings and destinations is defined by the **Dests** entry in the document's name dictionary (see Section 3.6.3, "Name Dictionary"). The value of this entry is a name tree (Section 3.8.5, "Name Trees") mapping name strings to destinations. (The keys in the name tree may be treated as text strings for display purposes.) The destination value associated with a key in the name tree may be either an array or a dictionary, as described in the preceding paragraph.

Note: *The use of strings as destination names is a PDF 1.2 feature. If compatibility with earlier versions of PDF is required, only name objects may be used to refer to named destinations. A document that supports PDF 1.2 can contain both types. However, if backward compatibility is not a consideration, applications should use the string form of representation in the* ***Dests*** *name tree.*

8.2.2 Document Outline

items by clicking them with the mouse. When an
dren in the hierarchy become visible on the sc
open or closed, selectively revealing or hiding
When an item is closed, all of its descendants in
ing the text of any visible item *activates* the item,
jump to a destination or trigger an action associ

The root of a document's outline hierarchy is a
the **Outlines** entry in the document catalog (see
log"). Table 8.3 shows the contents of this diction
within the hierarchy is defined by an *outline i*
items at each level of the hierarchy form a link
their **Prev** and **Next** entries and accessed throug
parent item (or in the outline dictionary in the c
played on the screen, the items at a given level a
occur in the linked list. (See also implementatio

TABLE 8.3 Entries in the outline dictionary

KEY	TYPE	VALUE
Type	name	*(Optional)* The type of PDF object that this dictionary describes; if present, must be **Outlines** for an outline dictionary.
First	dictionary	*(Required if there are any open or closed outline entries; must be an indirect reference)* An outline item dictionary representing the first top-level item in the outline.
Last	dictionary	*(Required if there are any open or closed outline entries; must be an indirect reference)* An outline item dictionary representing the last top-level item in the outline.
Count	integer	*(Required if the document has any open outline entries)* The total number of open items at all levels of the outline. This entry should be omitted if there are no open outline items.

TABLE 8.4 Entries in an outline item dictionary

K
T
P

KEY	TYPE	VALUE
Prev	dictionary	*(Required for all but the first i* The previous item at this outli
Next	dictionary	*(Required for all but the last i* The next item at this outline l
First	dictionary	*(Required if the item has any* first of this item's immediate c
Last	dictionary	*(Required if the item has any* last of this item's immediate c
Count	integer	*(Required if the item has any* ber of its open descendants a item is closed, a negative int descendants would appear if t
Dest	name, byte string, or array	*(Optional; not permitted if an* played when this item is activ implementation note 75 in Appendix H).
A	dictionary	*(Optional; PDF 1.1; not permitted if a **Dest** entry is present)* The action to be performed when this item is activated (see Section 8.5, "Actions").
SE	dictionary	*(Optional; PDF 1.3; must be an indirect reference)* The structure element to which the item refers (see Section 10.6.1, "Structure Hierarchy"). ***Note:** The ability to associate an outline item with a structure element (such as the beginning of a chapter) is a PDF 1.3 feature. For backward compatibility with earlier PDF versions, such an item should also specify a destination (**Dest**) corresponding to an area of a page where the contents of the designated structure element are displayed.*
C	array	*(Optional; PDF 1.4)* An array of three numbers in the range 0.0 to 1.0, representing the components in the **DeviceRGB** color space of the color to be used for the outline entry's text. Default value: [0.0 0.0 0.0].
F	integer	*(Optional; PDF 1.4)* A set of flags specifying style characteristics for display-

order). Table 8.5 shows the meanings of the f reserved and must be set to 0.

TABLE 8.5 Outline ite

BIT POSITION	NAME	MEANING
1	Italic	If set, display the item in italic.
2	Bold	If set, display the item in bold.

Example 8.1 shows a typical outline dictionary Appendix G for an example of a complete outlin

Example 8.1

```
21  0  obj
    <<  /Count  6
        /First  22 0 R
        /Last  29 0 R
    >>
endobj

22  0  obj
    <<  /Title  (Chapter 1)
        /Parent  21 0 R
        /Next  26 0 R
        /First  23 0 R
        /Last  25 0 R
        /Count  3
        /Dest  [3 0 R  /XYZ  0 792 0]
    >>
endobj
```

8.2.3 Thumbnail Images

A PDF document can define *thumbnail images* representing the contents of its pages in miniature form. A viewer application can display these images on the

The thumbnail image for a page is an image
entry in the page object (see "Page Objects" on
ture for an image dictionary (Section 4.8.4, "I
Width, **Height**, **ColorSpace**, **BitsPerComponent**,
cant; all of the other entries listed in Table 4.39 o
(If a **Subtype** entry is specified, its value must be
must be either **DeviceGray** or **DeviceRGB**, or an
these. Example 8.2 shows a typical thumbnail im

Example 8.2

```
12  0  obj
   <<  /Width  76
       /Height  99
       /ColorSpace  /DeviceRGB
       /BitsPerComponent  8
       /Length  13 0 R
       /Filter  [/ASCII85Decode  /DCTDecode]
   >>
stream
s4IA>!"M;*Ddm8XA,lT0!!3,S!/(=R!<E3%!<N<(!WrK*!WrN,
…Omitted data…
endstream
endobj

13  0  obj                                    % Length of stream
   …
endobj
```

8.2.4 Collections

Beginning with PDF 1.7, PDF documents can specify how a viewer application's user interface presents collections of file attachments, where the attachments are related in structure or content. Such a presentation is called a portable collection. The intent of *portable collections* is to present, sort, and search collections of related documents, such as email archives, photo collections, and engineering bid

A *collection dictionary* specifies the viewing and portable collections. If this dictionary is present terface presents the document as a portable colle tree specifies file attachments (see Section 3.10.3

When a PDF 1.7-compliant viewer application f taining a collection, it must display the content with a list of the documents present in the **Emb** ment list must include the additional document lection schema. The initial document can be t embedded documents.

The page content in the initial document typi helps the viewer understand what is contained and an introductory paragraph.

The file attachments comprising a collection ar name tree. All attachments in that tree are in th in that tree are not.

Table 8.6 describes the entries in a collection dictionary.

TABLE 8.6 Entries in a collection dictionary

KEY	TYPE	VALUE
Type	name	(*Optional*) The type of PDF object that this dictionary describes; if present, must be Collection for a collection dictionary.
Schema	dictionary	(*Optional*) A collection schema dictionary (see Table 8.7). If absent, the PDF viewer application may choose useful defaults that are known to exist in a file specification dictionary, such as the file name, file size, and modified date.
D	byte string	(*Optional*) A string that identifies an entry in the **EmbeddedFiles** name tree, controlling the document that is initially presented in the user interface. If the **D** entry is missing or in error, the initial

KEY	TYPE	VALUE
View	name	(*Optional*) The initi **D** The collecti information column for to the user. **T** The collecti in the collec formation fr top-level inf **H** The collecti the user fro Default value: **D**
Sort	dictionary	(*Optional*) A collecti which items in the c (see Table 8.9 on pa

A *collection schema dictionary* consists of a variable number of individual collection field dictionaries. Each collection field dictionary has a key chosen by the producer, which is used to associate a field with data in a file specification. Table 8.7 describes the entries in a collection schema dictionary.

TABLE 8.7 Entries in a collection schema dictionary

KEY	TYPE	VALUE
Type	name	(*Optional*) The type of PDF object that this dictionary describes; if present, must be CollectionSchema for a collection schema dictionary.
Other keys chosen by producer	dictionary	(*Optional*) Each dictionary entry is a collection field dictionary. Each key name is chosen at the discretion of the producer. The key name of each collection field dictionary is used to identify a corresponding collection item dictionary in a file specification dictio-

TABLE 8.8 Entries in a collection

KEY	TYPE	VALUE
Type	name	(*Optional*) The type of PD must be CollectionField for
Subtype	name	(*Required*) The subtype of tionary describes. This ent field. The following values ident collection subitem dictiona **S** A text field. The fie **D** A date field. The fi **N** A number field. Th The following values identi **F** The field data is th tified by the **UF** entry of the file specification, if present; otherwise by the **F** entry of the file specification (see Table 3.41). **Desc** The field data is the description of the embedded file stream, as identified by the **Desc** entry in the file specification dictionary (see Table 3.41). **ModDate** The field data is the modification date of the embedded file stream, as identified by the **ModDate** entry in the embedded file parameter dictionary (see Table 3.43). **CreationDate** The field data is the creation date of the embedded file stream, as identified by the **CreationDate** entry in the embedded file parameter dictionary (see Table 3.43). **Size** The field data is the size of the embedded file, as identified by the **Size** entry in the embedded file parameter dictionary (see Table 3.43).
N	text string	(*Required*) The textual field name that is displayed to the user by the PDF

KEY	TYPE	VALUE
V	boolean	(*Optional*) The initial visi value: true.
E	boolean	(*Optional*) A flag indicati provide support for editing

A *collection sort dictionary* identifies the fields t collection. The type of sorting depends on the ty

- Text strings are ordered lexically from smalle specified.
- Numbers are ordered numerically from small specified.
- Dates are ordered from oldest to newest, if asc

Table 8.9 describes the entries in a collection sor

TABLE 8.9 Entries in a collection sort dictionary

KEY	TYPE	VALUE
Type	name	(*Optional*) The type of PDF object that this dictionary describes; if present, must be CollectionSort for a collection sort dictionary.
S	name or array	(*Required*) The name or names of fields that the PDF viewer application uses to sort the items in the collection. If the value is a name, it identifies a field described in the parent collection dictionary. If the value is an array, each element of the array is a name that identifies a field described in the parent collection dictionary. The array form is used to allow additional fields to contribute to the sort, where each additional field is used to break ties. More specifically, if multiple collection item dictionaries have the same value for the first field named in the array, the values for successive fields named in the array are used for sorting, until a unique order is determined or until the named fields are exhausted.
A		

Example 8.3 shows a collection dictionary repr each item in the collection is an email message contained in file specification dictionaries. Th with each email is described in a collection sche nizational data (from, to, date, and subject) is p tionary, but the size data comes from the embed

Example 8.3

```
/Collection <<
    /Type /Collection
    /Schema <<
        /Type /CollectionSchema
        /from << /Subtype /S /N (From) /O 1 /V true /E
        /to << /Subtype /S /N (To) /O 2 /V true /E false
        /date << /Subtype /D /N (Date received) /O 3 /
        /subject << /Subtype /S /N (Subject) /O 4 /V tr
        /size << /Subtype /Size /N (Size) /O 5 /V true /
        >>
    /D (Doc1)
    /View /D
    /Sort << /S /date /A false >>
>>
```

Example 8.4 shows a collection item dictionary and a collection subitem dictionary. These dictionaries contain entries that correspond to the schema entries specified in Example 8.3. Section 3.10.5, "Collection Items" specifies the collection item and collection subitem dictionaries.

Example 8.4

```
/CI <<
    /Type /CollectionItem
    /from (Rob McAfee)
    /to (Patty McAfee)
    /subject <<
        /Type /CollectionSubitem
        /P (Re:)
```

8.3 Page-Level Navigation

This section describes PDF facilities that enable page within a document:

- *Page labels* for numbering or otherwise ident tion 8.3.1)
- *Article threads*, which chain together items of c are logically connected but not physically sequ
- *Presentations* that display the document in the from one page to the next either automaticall tion 8.3.3)

For another important form of page-level navig page 622.

8.3.1 Page Labels

Each page in a PDF document is identified by an integer *page index* that expresses the page's relative position within the document. In addition, a document may optionally define *page labels (PDF 1.3)* to identify each page visually on the screen or in print. Page labels and page indices need not coincide: the indices are fixed, running consecutively through the document starting from 0 for the first page, but the labels can be specified in any way that is appropriate for the particular document. For example, if the document begins with 12 pages of front matter numbered in roman numerals and the remainder of the document is numbered in arabic, the first page would have a page index of 0 and a page label of i, the twelfth page would have index 11 and label xii, and the thirteenth page would have index 12 and label 1.

For purposes of page labeling, a document can be divided into *labeling ranges*, each of which is a series of consecutive pages using the same numbering system. Pages within a range are numbered sequentially in ascending order. A page's label consists of a numeric portion based on its position within its labeling range,

number tree (Section 3.8.6, "Number Trees"), index of the first page in a labeling range. The c *dictionary* defining the labeling characteristics tree must include a value for page index 0. Table label dictionary. (See implementation note 76 in

Example 8.5 shows a document with pages label

i, ii, iii, iv, 1, 2, 3, A–8, A–9, …

Example 8.5

```
1 0 obj
   <<  /Type  /Catalog
       /PageLabels  << /Nums [  0  <<  /S  /r  >>
                                4  <<  /S  /D  >>
                                7  <<  /S  /D
                                       /P  (A–)
                                       /St  8
                                   >>
                             ]
                     >>
       …
   >>
endobj
```

TABLE 8.10 Entries in a page label dictionary

KEY	TYPE	VALUE
Type	name	*(Optional)* The type of PDF object that this dictionary describes; if present, must be **PageLabel** for a page label dictionary.
S	name	*(Optional)* The numbering style to be used for the numeric portion of each page label: D Decimal arabic numerals R Uppercase roman numerals r Lowercase roman numerals A Uppercase letters (A to Z for the first 26 pages, AA to ZZ for the next 26, and so on)

KEY	TYPE	VALUE
P	text string	*(Optional)* The label prefix for page labels i
St	integer	*(Optional)* The value of the numeric por sequent pages are numbered sequentially equal to 1. Default value: 1.

8.3.2 Articles

Some types of documents may contain sequenc cally connected but not physically sequential. Fo gin on the first page of a newsletter and nonconsecutive interior pages. To represent suc tiguous but logically related items, a PDF docum *cles (PDF 1.1)*. The sequential flow of an article i individual content items that make up the articl PDF viewer applications can provide navigation low a thread from one bead to the next.

The optional **Threads** entry in the document catalog (see Section 3.6.1, "Document Catalog") holds an array of *thread dictionaries* (Table 8.11) defining the document's articles. Each individual bead within a thread is represented by a *bead dictionary* (Table 8.12). The thread dictionary's **F** entry points to the first bead in the thread; the beads are chained together sequentially in a doubly linked list through their **N** (next) and **V** (previous) entries. In addition, for each page on which article beads appear, the page object (see "Page Objects" on page 144) should contain a **B** entry whose value is an array of indirect references to the beads on the page, in drawing order.

TABLE 8.11 Entries in a thread dictionary

KEY	TYPE	VALUE
Type	name	*(Optional)* The type of PDF object that this dictionary describes; if present, must be **Thread** for a thread dictionary.
F		
I		

TABLE 8.12 Entries in a bea

KEY	TYPE	VALUE
Type	name	*(Optional)* The type of PDF object th **Bead** for a bead dictionary.
T	dictionary	*(Required for the first bead of a thread; ence)* The thread to which this bead b
		Note: *In PDF 1.1, this entry is permitt and higher, it is permitted for any bead*
N	dictionary	*(Required; must be an indirect referenc* this entry points to the first.
V	dictionary	*(Required; must be an indirect referen* bead, this entry points to the last.
P	dictionary	*(Required; must be an indirect refere* which this bead appears.
R	rectangle	*(Required)* A rectangle specifying the location of this bead on the page.

Example 8.6 shows a thread with three beads.

Example 8.6

```
22 0 obj
   << /F 23 0 R
      /I << /Title (Man Bites Dog) >>
   >>
endobj

23 0 obj
   << /T 22 0 R
      /N 24 0 R
      /V 25 0 R
      /P 8 0 R
      /R [158 247 318 905]
```

```
24 0 obj
   << /T 22 0 R
      /N 25 0 R
      /V 23 0 R
      /P 8 0 R
      /R [322 246 486 904]
   >>
endobj

25 0 obj
   << /T 22 0 R
      /N 23 0 R
      /V 24 0 R
      /P 10 0 R
      /R [157 254 319 903]
   >>
endobj
```

8.3.3 Presentations

Some PDF viewer applications may allow a document to be displayed in the form of a *presentation* or slide show, advancing from one page to the next either automatically or under user control. In addition, PDF 1.5 introduces the ability to advance between different states of the same page (see "Sub-page Navigation" on page 601).

Note: *PDF 1.4 introduces a different mechanism, known as* alternate presentations*, for slide show displays, described in Section 9.4, "Alternate Presentations."*

A page object (see "Page Objects" on page 144) may contain two optional entries, **Dur** and **Trans** *(PDF 1.1)*, to specify how to display that page in presentation mode. The **Trans** entry contains a *transition dictionary* describing the style and duration of the visual transition to use when moving from another page to the given page during a presentation. Table 8.13 shows the contents of the transition dictionary. (Some of the entries shown are needed only for certain transition styles, as indicated in the table.)

TABLE 8.13 Entries in a transit

KEY	TYPE	VALUE
Type	name	*(Optional)* The type of PDF object tha **Trans** for a transition dictionary.
S	name	*(Optional)* The *transition style* to use w presentation. Default value: R.
		Split — Two lines sweep across t be either horizontal or v the page or outward fro entries, respectively.
		Blinds — Multiple lines, evenly sp the same direction to rev zontal or vertical, as spe downward; vertical lines
		Box — A rectangular box sweep from the center, as specified by the **M** entry, revealing the new page.
		Wipe — A single line sweeps across the screen from one edge to the other in the direction specified by the **Di** entry, revealing the new page.
		Dissolve — The old page dissolves gradually to reveal the new one.
		Glitter — Similar to Dissolve, except that the effect sweeps across the page in a wide band moving from one side of the screen to the other in the direction specified by the **Di** entry.
		R — The new page simply replaces the old one with no special transition effect; the **D** entry is ignored.
		Fly — *(PDF 1.5)* Changes are flown out or in (as specified by **M**), in the direction specified by **Di**, to or from a location that is offscreen except when **Di** is None.
		Push — *(PDF 1.5)* The old page slides off the screen while the new page slides in, pushing the old page out in the direction specified by **Di**.
		Cover — *(PDF 1.5)* The new page slides on to the screen in the direction specified

KEY	TYPE	VALUE
D	number	*(Optional)* The duration of the transitio
Dm	name	*(Optional; Split and Blinds transition sty* transition effect occurs: H Horizontal V Vertical Default value: H.
M	name	*(Optional; Split, Box and Fly transition st* fied transition effect: I Inward from the edge O Outward from the ce Default value: I.
Di	number or name	*(Optional; Wipe, Glitter, Fly, Cover, Uncov* in which the specified transition effect starting from a left-to-right direction. (which is measured clockwise from the top.) The following numeric values are valid: 0 Left to right 90 Bottom to top (Wipe only) 180 Right to left (Wipe only) 270 Top to bottom 315 Top-left to bottom-right (Glitter only) The only valid name value is None, which is relevant only for the Fly transition when the value of SS is not 1.0. Default value: 0.
SS	number	(*Optional; PDF 1.5; Fly transition style only*) The starting or ending scale at which the changes are drawn. If **M** specifies an inward transition, the scale of the changes drawn progresses from **SS** to 1.0 over the course of the transition. If **M** specifies an outward transition, the scale of the changes drawn progresses from 1.0 to **SS** over the course of the transition
B		
—		

sition duration specified for a page (page 2 in the that page from another page; the transition *from* page's transition duration.

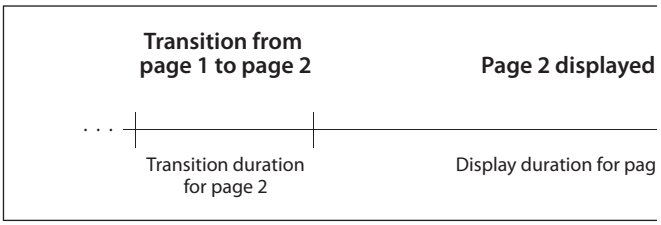


FIGURE 8.1 *Presentation*

Example 8.7 shows the presentation parameters seconds. Before the page is displayed, there is two vertical lines sweep outward from the center

Example 8.7

```
10  0  obj
   <<  /Type  /Page
       /Parent  4 0 R
       /Contents  16 0 R
       /Dur  5
       /Trans  <<  /Type  /Trans
                   /D  3.5
                   /S  /Split
                   /Dm  /V
                   /M  /O
               >>
   >>
endobj
```

Sub-page Navigation

Note: *Viewer applications should save the state o user enters presentation mode and restore it when sures, for example, that transient changes to bulle document.*

A navigation node dictionary (see Table 8.14) s the user makes a navigation request; for exampl navigation nodes on a page form a doubly linke **Prev** entries. The primary node on a page is deter entry in a page dictionary (see Table 3.27).

Note: *It is recommended that a viewer applicati when in presentation mode (see Section 8.3.3, "Pr*

TABLE 8.14 Entries in a navigatio

KEY	TYPE	VALUE
Type	name	*(Optional)* The type of PDF object t for a navigation node dictionary.
NA	dictionary	*(Optional)* The sequence of actions to execute when a user navigates forward.
PA	dictionary	*(Optional)* The sequence of actions to execute when a user navigates backward.
Next	dictionary	*(Optional)* The next navigation node, if any.
Prev	dictionary	*(Optional)* The previous navigation node, if any.
Dur	number	*(Optional)* The maximum number of seconds before the viewer application should automatically advance forward to the next navigation node. If this entry is not specified, no automatic advance should occur.

A viewer application should support the notion of a *current* navigation node. When a user navigates to a page, if the page dictionary has a **PresSteps** entry, the node specified by that entry becomes the current node. (Otherwise, there is no current node.) If there is a request to navigate forward (such as an arrow key press) and there is a current navigation node, the following occurs:

2. The node specified by **Next** (if present) beco node.

Similarly, if there is a request to navigate backwa tion node, the following occurs:

1. The sequence of actions specified by **PA** (if pr

   Note: *If **PA** specifies an action that navigates scribed below for navigating to another page t present.*

2. The node specified by **Prev** (if present) beco node.

When navigating between nodes, it is possible t effects are similar to the page transitions specifi ever, they use a different mechanism; see "Transi

Note: *"Forward" and "backward" are determined by user actions, such as pressing right or left arrow keys, not by the actual page that is the destination of an action.*

If there is a request to navigate to another page (regardless of whether there is a current node) and that page's dictionary contains a **PresSteps** entry, the following occurs:

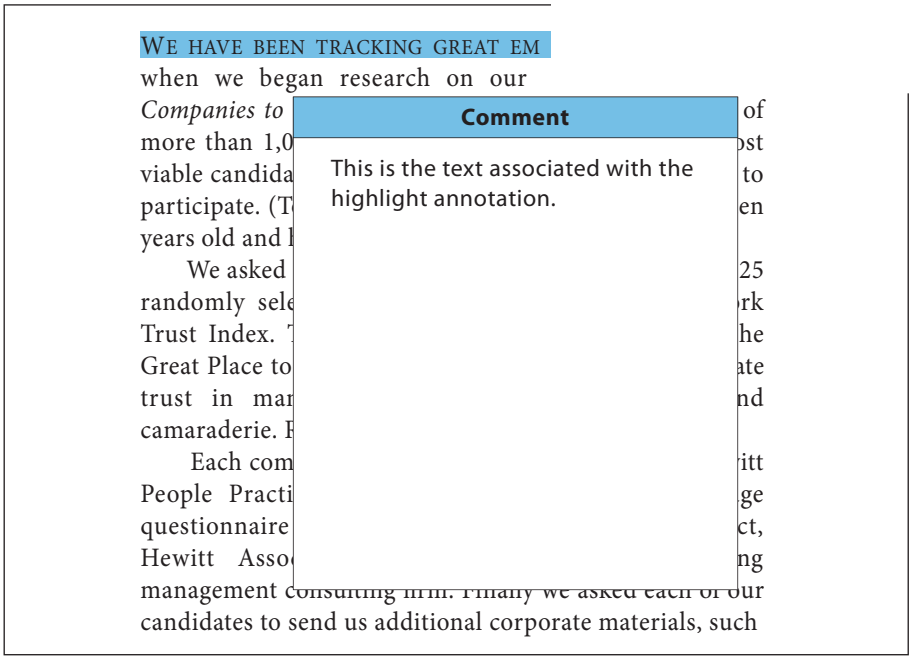
1. The navigation node represented by **PresSteps** becomes the current node.
2. If the navigation request was forward, or if the navigation request was for random access (such as by clicking on a link), the actions specified by **NA** are executed and the node specified by **Next** becomes the new current node, as described above.

   If the navigation request was backward, the actions specified by **PA** are executed and the node specified by **Prev** becomes the new current node, as described above.

8.4 Annotations

An *annotation* associates an object such as a note on a page of a PDF document, or provides a means of the mouse and keyboard. PDF includ notation types, described in detail in Section 8.4.

Many of the standard annotation types may be d *closed* state. When closed, they appear on the pa as an icon, a box, or a rubber stamp, depending When the user *activates* the annotation by clicki ject, such as by opening a pop-up window displa playing a sound or a movie.



gation order explicit with the optional **Tabs** entry The following are the possible values for this ent

- R (row order): Annotations are visited in rows page. The direction within a row is determin viewer preferences dictionary (see Section 8.1, annotation visited is the first annotation in the row is encountered, the first annotation in the
- C (column order): Annotations are visited in and down the page. Columns are ordered by t preferences dictionary (see Section 8.1, "View tation visited is the one at the top of the first umn is encountered, the first annotation in th
- S (structure order): Annotations are visited in t the structure tree (see Section 10.6, "Logical St tions that are not included in the structure tre

Note: *The descriptions above assume the page is being viewed in the orientation specified by the* ***Rotate*** *entry.*

The behavior of each annotation type is implemented by a software module called an *annotation handler*. Handlers for the standard annotation types are built directly into the PDF viewer application; handlers for additional types can be supplied as plug-in extensions.

8.4.1 Annotation Dictionaries

The optional **Annots** entry in a page object (see "Page Objects" on page 144) holds an array of *annotation dictionaries*, each representing an annotation associated with the given page. Table 8.15 shows the required and optional entries that are common to all annotation dictionaries. The dictionary may contain additional entries specific to a particular annotation type; see the descriptions of individual annotation types in Section 8.4.5, "Annotation Types," for details.

TABLE 8.15 Entries common to all an

KEY	TYPE	VALUE
Type	name	*(Optional)* The type of PDF o must be **Annot** for an annotatio
Subtype	name	*(Required)* The type of annotati on page 615 for specific values.
Rect	rectangle	*(Required)* The *annotation recta* the page in default user space u
Contents	text string	*(Optional)* Text to be displayed does not display text, an altern human-readable form. In eithe document's contents in support other purposes (see Section 10. "Annotation Types" for more d notation type.
P	dictionary	*(Optional; PDF 1.3; not used in FDF files)* An indirect reference to the page object with which this annotation is associated. ***Note:*** *This entry is required for screen annotations associated with rendition actions (PDF 1.5; see "Screen Annotations" on page 639 and "Rendition Actions" on page 668).*
NM	text string	*(Optional; PDF 1.4)* The *annotation name*, a text string uniquely identifying it among all the annotations on its page.
M	date or text string	*(Optional; PDF 1.1)* The date and time when the annotation was most recently modified. The preferred format is a date string as described in Section 3.8.3, "Dates," but viewer applications should be prepared to accept and display a string in any format. (See implementation note 78 in Appendix H.)
F	integer	*(Optional; PDF 1.1)* A set of flags specifying various characteristics of the annotation (see Section 8.4.2, "Annotation Flags"). Default value: 0.
AP	dictionary	*(Optional; PDF 1.2)* An *appearance dictionary* specifying how the annotation is

KEY	TYPE	VALUE
AS	name	*(Required if the appearance dict PDF 1.2)* The annotation's *ap* appearance stream from an ap pearance Streams" and also imp
Border	array	*(Optional)* An array specifying The border is specified as a rou In PDF 1.0, the array consists o radius, vertical corner radius, a If the corner radii are 0, the b border width is 0, no border is dix H.) In PDF 1.1, the array may ha defining a pattern of dashes an dash array is specified in the sa of the graphics state (see "Line **der** value of `[0 0 1 [3 2]]` spec drawn with 3-unit dashes alternating with 2-unit gaps. Note that no dash phase is specified; the phase is assumed to be 0. (See implementation note 82 in Appendix H.) ***Note:*** *In PDF 1.2 or later, this entry may be ignored in favor of the* ***BS*** *entry (see above); see implementation note 82 in Appendix H.* Default value: `[0 0 1]`.
C	array	*(Optional; PDF 1.1)* An array of numbers in the range 0.0 to 1.0, representing a color used for the following purposes: • The background of the annotation's icon when closed • The title bar of the annotation's pop-up window • The border of a link annotation The number of array elements determines the color space in which the color is defined:

KEY	TYPE	VALUE
StructParent	integer	*(Required if the annotation is a s* of the annotation's entry in the s ements from Content Items" on
OC	dictionary	*(Optional; PDF 1.5)* An option ship dictionary (see Section 4.1 content properties for the anno bility is determined based on th in the **F** entry (see Section 8.4. invisible, the annotation is skip

8.4.2 Annotation Flags

The value of the annotation dictionary's **F** entry i taining flags specifying various characteristics within the flag word are numbered from 1 (low 8.16 shows the meanings of the flags; all undefin be set to 0.

TABLE 8.16 Annotation flags

BIT POSITION	NAME	MEANING
1	Invisible	If set, do not display the annotation if it does not belong to one of the standard annotation types and no annotation handler is available. If clear, display such an unknown annotation using an appearance stream specified by its appearance dictionary, if any (see Section 8.4.4, "Appearance Streams").
2	Hidden	*(PDF 1.2)* If set, do not display or print the annotation or allow it to interact with the user, regardless of its annotation type or whether an annotation handler is available. In cases where screen space is limited, the ability to hide and show annotations selectively can be used in combination with appearance streams (see Section 8.4.4, "Appearance Streams") to display auxiliary pop-up information similar in function to online help systems. (See implementation note 83 in Appendix H.)
3		

BIT POSITION	NAME	MEANING
4	NoZoom	*(PDF 1.3)* If set, do not scale t fication of the page. The loca the upper-left corner of its an the page magnification. See be
5	NoRotate	*(PDF 1.3)* If set, do not rotate tion of the page. The upper-lef a fixed location on the page, re ther discussion.
6	NoView	*(PDF 1.3)* If set, do not displ interact with the user. The a setting of the Print flag) but s screen display and user intera
7	ReadOnly	*(PDF 1.3)* If set, do not allow annotation may be displayed NoView and Print flags) but s appearance in response to mouse motions. ***Note:*** *This flag is ignored for widget annotations; its function is subsumed by the ReadOnly flag of the associated form field (see Table 8.70 on page 676).*
8	Locked	*(PDF 1.4)* If set, do not allow the annotation to be deleted or its properties (including position and size) to be modified by the user. However, this flag does not restrict changes to the annotation's contents, such as the value of a form field. (See implementation note 84 in Appendix H.)
9	ToggleNoView	*(PDF 1.5)* If set, invert the interpretation of the NoView flag for certain events. A typical use is to have an annotation that appears only when a mouse cursor is held over it; see implementation note 85 in Appendix H.
10	LockedContents	*(PDF 1.7)* If set, do not allow the contents of the annotation to be modified by the user. This flag does not restrict deletion of the annotation or changes to other annotation properties, such as position and size.

If the NoZoom flag is set, the annotation always maintains the same fixed size on

In either case, the annotation's position is deter
upper-left corner of its annotation rectangle, as
annotation dictionary and interpreted in the def
the default user space is scaled or rotated, the po
of the annotation rectangle are different in the a
in the original user space. The viewer applicati
matically. However, it does not actually change t
continues to describe the annotation's relations
user space.

For example, Figure 8.3 shows how an annotati
mains upright when the page it is on is rotated
left corner of the annotation remains at the sam
annotation pivots around that point.

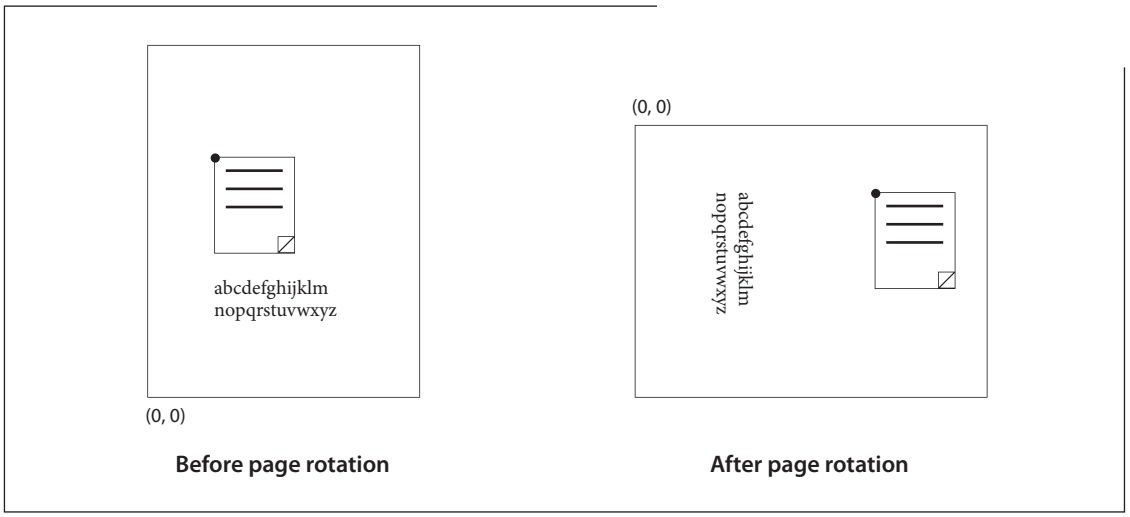


FIGURE 8.3 *Coordinate adjustment with the NoRotate flag*

8.4.3 Border Styles

PDF 1.2, some types of annotations may instead
tics in a *border style dictionary* designated by the
tionaries are also used to specify the width and
by line, square, circle, and ink annotations. Table
the border style dictionary. If neither the **Borde**
border is drawn as a solid line with a width of 1

TABLE 8.17 Entries in a border

KEY	TYPE	VALUE
Type	name	*(Optional)* The type of PDF object th **Border** for a border style dictionary.
W	number	*(Optional)* The border width in point value: 1.
S	name	*(Optional)* The border style: S (Solid) A solid rectangle surr D (Dashed) A dashed rectangle surrounding the annotation. The dash pattern is specified by the **D** entry (see below). B (Beveled) A simulated embossed rectangle that appears to be raised above the surface of the page. I (Inset) A simulated engraved rectangle that appears to be recessed below the surface of the page. U (Underline) A single line along the bottom of the annotation rectangle. Other border styles may be defined in the future. Default value: S.
D	array	*(Optional)* A *dash array* defining a pattern of dashes and gaps to be used in drawing a dashed border (border style D above). The dash array is specified in the same format as in the line dash pattern parameter of the graphics state (see "Line Dash Pattern" on page 217). The dash phase is not specified and is assumed to be 0. For example, a **D** entry of [3 2] specifies a border drawn with 3-point dashes alternating with 2-point gaps. Default value: [3].

TABLE 8.18 Entries in a border e

KEY	TYPE	VALUE
S	name	*(Optional)* A name representing the b S No effect: the border is as des C The border should appear "cl are honored. Default value: S.
I	number	*(Optional; valid only if the value of **S** is* fect. Suggested values range from 0 to

8.4.4 Appearance Streams

Beginning with PDF 1.2, an annotation can s *streams* as an alternative to the simple border a in earlier versions. Appearance streams enable the annotation to be presented visually in different ways to reflect its interactions with the user. Each appearance stream is a form XObject (see Section 4.9, "Form XObjects"): a self-contained content stream to be rendered inside the annotation rectangle.

The following method is used to map from the coordinate system of the appearance XObject (as defined by its **Matrix** entry; see Table 4.45) to the annotation's rectangle in default user space:

Algorithm 8.1

1. The appearance's bounding box (specified by its **BBox** entry) is transformed, using **Matrix**, to produce a quadrilateral with arbitrary orientation. The *transformed appearance box* is the smallest upright rectangle that encompasses this quadrilateral.
2. A matrix *A* is computed that scales and translates the transformed appearance box to align with the edges of the annotation's rectangle (specified by the **Rect** entry). *A* maps the lower-left corner (the corner with the smallest x and y coordinates) and the upper-right corner (the corner with the greatest x and y coordinates) of the

The annotation may be further scaled and ro
NoRotate flag is set (see Section 8.4.2, "Annotat
applied to the annotation as a whole is also appli

In PDF 1.4, an annotation appearance can incl
ance's stream dictionary does not contain a **Grou**
lated, non-knockout transparency group. Other
values specified in the group dictionary (see Sec
XObjects") are used.

The transparency group is composited with a
content along with any previously painted ann
Normal, an alpha constant of 1.0, and a soft ma
note 87 in Appendix H.)

Note: *If a transparent annotation appearance is p*
drawn without using an appearance stream, the
dent. This is because such annotations are someti
conform to the Adobe imaging model. Also, the effect of highlighting a transparent annotation appearance is implementation-dependent.

An annotation can define as many as three separate appearances:

- The *normal appearance* is used when the annotation is not interacting with the user. This appearance is also used for printing the annotation.
- The *rollover appearance* is used when the user moves the cursor into the annotation's active area without pressing the mouse button.
- The *down appearance* is used when the mouse button is pressed or held down within the annotation's active area.

Note: *As used here, the term* mouse *denotes a generic pointing device that controls the location of a cursor on the screen and has at least one button that can be pressed, held down, and released. See Section 8.5.2, "Trigger Events," for further discussion.*

TABLE 8.19 Entries in an appear

KEY	TYPE	VALUE
N	stream or dictionary	*(Required)* The annotation's n
R	stream or dictionary	*(Optional)* The annotation's r the **N** entry.
D	stream or dictionary	*(Optional)* The annotation's d **N** entry.

Each entry in the appearance dictionary may c stream or an *appearance subdictionary*. In the l fines multiple appearance streams correspondin the annotation.

For example, an annotation representing an inter appearance states named On and Off. Its appeara as

```
/AP << /N << /On formXObject1
             /Off formXObject2
          >>
       /D << /On formXObject3
             /Off formXObject4
          >>
   >>
```

where $formXObject_1$ and $formXObject_2$ define the check box's normal appearance in its checked and unchecked states, and $formXObject_3$ and $formXObject_4$ provide visual feedback, such as emboldening its outline, when the user clicks it. (No **R** entry is defined because no special appearance is needed when the user moves the cursor over the check box without pressing the mouse button.) The choice between the checked and unchecked appearance states is determined by the **AS** entry in the annotation dictionary (see Table 8.15 on page 606).

havior (such as displaying nothing) if an annota
pearance state for which no appearance is defined

For convenience in managing appearance stream
entry in a PDF document's name dictionary (se
ary") can contain a name tree mapping name st
name strings have no standard meanings; no
streams by name.

8.4.5 Annotation Types

PDF supports the standard annotation types lis
sections describe each of these types in detail.
annotation types, and further standard types ma
plementation note 88 in Appendix H.)

The values in the first column of Table 8.20 repr
dictionary's **Subtype** entry. The third column in
a *markup annotation*, as described in "Markup Annotations," below. The section also provides more information about the value of the **Contents** entry for different annotation types.

TABLE 8.20 Annotation types

ANNOTATION TYPE	DESCRIPTION	MARKUP?	DISCUSSED IN SECTION
Text	Text annotation	Yes	"Text Annotations" on page 621
Link	Link annotation	No	"Link Annotations" on page 622
FreeText	*(PDF 1.3)* Free text annotation	Yes	"Free Text Annotations" on page 623
Line	*(PDF 1.3)* Line annotation	Yes	"Line Annotations" on page 626
Square	*(PDF 1.3)* Square annotation	Yes	"Square and Circle Annotations" on page 630
Circle	*(PDF 1.3)* Circle annotation	Yes	"Square and Circle Annotations" on page 630
P			
P			

ANNOTATION TYPE	DESCRIPTION	MARKUP?	
Highlight	*(PDF 1.3)* Highlight annotation	Yes	
Underline	*(PDF 1.3)* Underline annotation	Yes	
Squiggly	*(PDF 1.4)* Squiggly-underline annotation	Yes	
StrikeOut	*(PDF 1.3)* Strikeout annotation	Yes	
Stamp	*(PDF 1.3)* Rubber stamp annotation	Yes	
Caret	*(PDF 1.5)* Caret annotation	Yes	
Ink	*(PDF 1.3)* Ink annotation	Yes	
Popup	*(PDF 1.3)* Pop-up annotation	No	
FileAttachment	*(PDF 1.3)* File attachment annotation	Yes	
Sound	*(PDF 1.2)* Sound annotation	Yes	"Sound Annotations" on page 638
Movie	*(PDF 1.2)* Movie annotation	No	"Movie Annotations" on page 639
Widget	*(PDF 1.2)* Widget annotation	No	"Widget Annotations" on page 640
Screen	*(PDF 1.5)* Screen annotation	No	"Screen Annotations" on page 639
PrinterMark	*(PDF 1.4)* Printer's mark annotation	No	"Printer's Mark Annotations" on page 643
TrapNet	*(PDF 1.3)* Trap network annotation	No	"Trap Network Annotations" on page 643
Watermark	*(PDF 1.6)* Watermark annotation	No	"Watermark Annotations" on page 644
3D	*(PDF 1.6)* 3D annotation	No	"3D Annotations" on page 791

Markup Annotations

As mentioned in Section 8.4.1, "Annotation Dictionaries", the meaning of an an-

Many annotation types are defined as *markup a*
primarily to mark up PDF documents (see Tab
text that appears as part of the annotation and
a viewer application, such as in a Comments pan

Markup annotations can be divided into the foll

- Free text annotations display text directly **Contents** entry specifies the displayed text.
- Most other markup annotations have an asso contain text. The annotation's **Contents** entry when the pop-up window is opened. These polygon, polyline, highlight, underline, squig stamp, caret, ink, and file attachment annotati
- Sound annotations do not have a pop-up win text specified by the **Contents** entry.

Note: *When separating text into paragraphs, a carriage return should be used (and not, for example, a line feed character).*

Note: *A subset of markup annotations are called* text markup annotations *(see "Text Markup Annotations" on page 633).*

The remaining annotation types are not considered markup annotations:

- The pop-up annotation type typically does not appear by itself; it is associated with a markup annotation that uses it to display text.

  Note: *The* ***Contents*** *entry for a pop-up annotation is relevant only if it has no parent; in that case, it represents the text of the annotation.*

- For all other annotation types (**Link**, **Movie**, **Widget**, **PrinterMark**, and **TrapNet**), the **Contents** entry provides an alternate representation of the annotation's contents in human-readable form, which is useful when extracting the document's contents in support of accessibility to users with disabilities or for other pur-

TABLE 8.21 Additional entries specific t

KEY	TYPE	VALUE
T	text string	*(Optional; PDF 1.1)* The text lab tion's pop-up window when ope the user who added the annotati
Popup	dictionary	*(Optional; PDF 1.3)* An indirect r editing the text associated with t
CA	number	*(Optional; PDF 1.4)* The constant tation (see Sections 7.1, "Over Opacity Computations"). This va tion in its closed state (including up window that appears when th The specified value is not used if Section 8.4.4, "Appearance Strea specify any transparency. (How appearance stream, it may incorp The implicit blend mode (see Section 7.2.4, "Blend Mode") is **Normal**. Default value: 1.0. ***Note:*** *If no explicit appearance stream is defined for the annotation, it is painted by implementation-dependent means that do not necessarily conform to the Adobe imaging model; in this case, the effect of this entry is implementation-dependent as well.*
RC	text string or text stream	*(Optional; PDF 1.5)* A rich text string (see "Rich Text Strings" on page 680) to be displayed in the pop-up window when the annotation is opened.
CreationDate	date	*(Optional; PDF 1.5)* The date and time (Section 3.8.3, "Dates") when the annotation was created.
IRT	dictionary	*(Required if an **RT** entry is present, otherwise optional; PDF 1.5)* A reference to the annotation that this annotation is "in reply to." Both annotations must be on the same page of the document. The relationship between the two annotations is specified by the **RT** entry.

KEY	TYPE	VALUE
Subj	text string	*(Optional; PDF 1.5)* Text repres addressed by the annotation.
RT	name	*(Optional; meaningful only if **IRT*** tionship (the "reply type") betwe id values are: **R** The annotation is c **IRT**. Viewer applicati individually but tog **Group** The annotation is gr discussion below. Default value: **R**.
IT	name	*(Optional; PDF 1.6)* A name desc tents allow viewer applications t iors of a single markup annotatio the same as the annotation type, t behave in a generic manner in a viewer application. Free text annotations (Table 8.25), line annotations (Table 8.26), polygon annotations (Table 8.29), and (in PDF 1.7) polyline annotations (Table 8.29) have defined intents, whose values are enumerated in the corresponding tables.
ExData	dictionary	*(Optional; PDF 1.7)* An *external data dictionary* specifying data to be associated with the annotation. This dictionary contains the following entries: **Type** *(optional):* If present, must be **ExData**. **Subtype** *(required):* a name specifying the type of data that the markup annotation is associated with. In PDF 1.7, the only defined value is **Markup3D**. For each value of **Subtype**, other entries are defined. Table 9.48 on page 835 lists the values that correspond to a subtype of **Markup3D**. (See also implementation note 96 in Appendix H.)

In PDF 1.6, a set of annotations can be grouped so that they function as a single unit when a user interacts with them. The group consists of a *primary annotation*,

nate annotations should be ignored. These entri
M, **C**, **T**, **Popup**, **CreationDate**, **Subj**, and **Open**.
annotation in a group, such as movement, cut,
viewer applications as acting on the entire group

Note: *A primary annotation may have replies that*
that is, that do not have an ***RT*** *value of* ***Group****.*

Annotation States

Beginning with PDF 1.5, annotations may have a
with them. The state is not specified in the anno
annotation that refers to the original annotation
entry (see Table 8.24). States are grouped into a
in Table 8.22.

TABLE 8.22 Annotation

STATE MODEL	STATE	DESCRIPTION
Marked	Marked	The annotation has been marked by the user.
	Unmarked	The annotation has not been marked by the user (the default).
Review	Accepted	The user agrees with the change.
	Rejected	The user disagrees with the change.
	Cancelled	The change has been cancelled.
	Completed	The change has been completed.
	None	The user has indicated nothing about the change (the default).

Annotations can be thought of as initially being in the default state for each state model. State changes made by a user are indicated in a text annotation with the following entries:

- **T**

Additional state changes are made by adding tex
vious reply for a given user.

Text Annotations

A *text annotation* represents a "sticky note" attac
ment. When closed, the annotation appears as
pop-up window containing the text of the note
viewer application. Text annotations do not scal
behave as if the NoZoom and NoRotate annotat
608) were always set. Table 8.23 shows the anno
to this type of annotation.

TABLE 8.23 Additional entries specifi

KEY	TYPE	VALUE
Subtype	name	*(Required)* The type of annotati for a text annotation.
Open	boolean	*(Optional)* A flag specifying whether the annotation should initially be displayed open. Default value: **false** (closed).
Name	name	*(Optional)* The name of an icon to be used in displaying the annotation. Viewer applications should provide predefined icon appearances for at least the following standard names: Comment Key Note Help NewParagraph Paragraph Insert Additional names may be supported as well. Default value: Note. ***Note:*** *The annotation dictionary's* ***AP*** *entry, if present, takes precedence over the* ***Name*** *entry; see Table 8.15 on page 606 and Section 8.4.4, "Appearance Streams."*
State	text string	*(Optional; PDF 1.5)* The state to which the original annotation should be set; see "Annotation States," above.

S

—

Example 8.8 shows the definition of a text annot

Example 8.8

```
22 0 obj
   << /Type /Annot
      /Subtype /Text
      /Rect [266 116 430 204]
      /Contents (The quick brown fox ate the lazy
   >>
endobj
```

Link Annotations

A *link annotation* represents either a hypertext l the document (see Section 8.2.1, "Destinations (Section 8.5, "Actions"). Table 8.24 shows the a cific to this type of annotation.

TABLE 8.24 Additional entries specific to a link annotation

KEY	TYPE	VALUE
Subtype	name	*(Required)* The type of annotation that this dictionary describes; must be **Link** for a link annotation.
A	dictionary	*(Optional; PDF 1.1)* An action to be performed when the link annotation is activated (see Section 8.5, "Actions").
Dest	array, name or byte string	*(Optional; not permitted if an **A** entry is present)* A destination to be displayed when the annotation is activated (see Section 8.2.1, "Destinations"; see also implementation note 89 in Appendix H).
H	name	*(Optional; PDF 1.2)* The annotation's *highlighting mode*, the visual effect to be used when the mouse button is pressed or held down inside its active area: N (None) No highlighting. I (Invert) Invert the contents of the annotation rectangle. O (Outline) Invert the annotation's border.

KEY	TYPE	VALUE
PA	dictionary	*(Optional; PDF 1.3)* A URI act associated with this annotation ture") changes an annotation f on page 654), it uses this entry that it can be changed back in c quently deleted.
QuadPoints	array	*(Optional; PDF 1.6)* An array of quadrilaterals in default user s should be activated. The coordi $x_1\ y_1\ x_2\ y_2\ x_3\ y_3\ x_4\ y_4$ specifying the four vertices of t orientation purposes, such as bottom of a quadrilateral is the If this entry is not present or t region specified by the **Rect** e nored if any coordinate in the a

Example 8.9 shows a link annotation that jumps to a destination elsewhere in the document.

Example 8.9

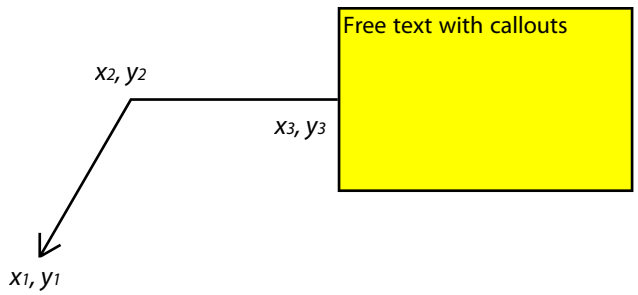
```
93 0 obj
   <<  /Type  /Annot
       /Subtype  /Link
       /Rect  [71  717  190  734]
       /Border  [16  16  1]
       /Dest  [3 0 R  /FitR  –4  399  199  533]
   >>
endobj
```

Free Text Annotations

TABLE 8.25 Additional entries specific t

KEY	TYPE	VALUE
Subtype	name	*(Required)* The type of annota **FreeText** for a free text annotatio
DA	string	*(Required)* The default appearan "Variable Text" on page 677). ***Note:*** *The annotation dictionary's entry; see Table 8.15 on page 606*
Q	integer	*(Optional; PDF 1.4)* A code speci used in displaying the annotatio 0 Left-justified 1 Centered 2 Right-justified Default value: 0 (left-justified).
RC	text string or text stream	*(Optional; PDF 1.5)* A rich text string (see "Rich Text Strings" on page 680) to be used to generate the appearance of the annotation.
DS	text string	*(Optional; PDF 1.5)* A default style string, as described in "Rich Text Strings" on page 680.
CL	array	*(Optional; PDF 1.6)* An array of four or six numbers specifying a callout line attached to the free text annotation. Six numbers [x_1 y_1 x_2 y_2 x_3 y_3] represent the starting, knee point, and ending coordinates of the line in default user space, as shown in Figure 8.4. Four numbers [x_1 y_1 x_2 y_2] represent the starting and ending coordinates of the line.
IT	name	*(Optional; PDF 1.6)* A name describing the intent of the free text annotation (see also Table 8.21). Valid values are FreeTextCallout, which means that the annotation is intended to function as a callout, and FreeTextTypeWriter, which means that the annotation is intended to function as a click-to-type or typewriter object.
BE	dictionary	(*Optional; PDF 1.6*) A border effect dictionary (see Table 8.18) used in conjunc-

KEY	TYPE	VALUE
RD	rectangle	*(Optional; PDF 1.6)* A set of four between two rectangles: the **Rec** tained within that rectangle. Th should be displayed. Any border **BE** entries, respectively, are appli The four numbers correspond t the left, top, right, and bottom co gle, respectively. Each value mus top and bottom differences must the left and right differences mus
BS	dictionary	*(Optional; PDF 1.6)* A border styl fying the line width and dash pat der. ***Note:*** *The annotation dictionary's* ***kList*** *and* ***BS*** *entries; see Table 8. Streams."*
LE	array	*(Optional; PDF 1.6)* An array of two names specifying the line ending styles to be used in drawing the annotation's border. The first and second elements of the array specify the line ending styles for the endpoints defined, respectively, by the first and second pairs of coordinates, (x_1, y_1) and (x_2, y_2), in the **L** array. Table 8.27 shows the possible values. Default value: [/None /None].



Line Annotations

A *line annotation (PDF 1.3)* displays a single s
opened, it displays a pop-up window containin
Table 8.26 shows the annotation dictionary entr
tation.

TABLE 8.26 Additional entries specifi

KEY	TYPE	VALUE
Subtype	name	*(Required)* The type of annotati for a line annotation.
L	array	*(Required)* An array of four num ending coordinates of the line in ***Note:*** *If the* ***LL*** *entry is present, t lines rather than the endpoints of*
BS	dictionary	*(Optional)* A border style dictionary (see Table 8.17 on page 611) specifying the width and dash pattern to be used in drawing the line. ***Note:*** *The annotation dictionary's* ***AP*** *entry, if present, takes precedence over the* ***L*** *and* ***BS*** *entries; see Table 8.15 on page 606 and Section 8.4.4, "Appearance Streams."*
LE	array	*(Optional; PDF 1.4)* An array of two names specifying the line ending styles to be used in drawing the line. The first and second elements of the array specify the line ending styles for the endpoints defined, respectively, by the first and second pairs of coordinates, (x_1, y_1) and (x_2, y_2), in the **L** array. Table 8.27 shows the possible values. Default value: [/None /None].
IC	array	*(Optional; PDF 1.4)* An array of numbers in the range 0.0 to 1.0 specifying the *interior color* with which to fill the annotation's line endings (see Table 8.27). The number of array elements determines the color space in which the color is defined: 0 No color; transparent 1 **DeviceGray** 3 **DeviceRGB**

KEY	TYPE	VALUE
LL	number	*(Required if **LLE** is present, otherw* in default user space that extend f the line itself, as shown in Figure appear in the direction that is cl ing point to its ending point (as s posite direction. Default value: 0 (no leader lines).
LLE	number	*(Optional; PDF 1.6)* A non-nega *line extensions* that extend from lines, as shown in Figure 8.5. Default value: 0 (no leader line e
Cap	boolean	*(Optional; PDF 1.6)* If **true**, the should be replicated as a caption ure 8.6 and Figure 8.7. The text the content, taking into account f Default value: false.
IT	name	*(Optional; PDF 1.6)* A name describing the intent of the line annotation (see also Table 8.21). Valid values are **LineArrow**, which means that the annotation is intended to function as an arrow, and **LineDimension**, which means that the annotation is intended to function as a dimension line.
LLO	number	(*Optional; PDF 1.7*) A non-negative number representing the length of the leader line offset, which is the amount of empty space between the endpoints of the annotation and the beginning of the leader lines.
CP	name	(*Optional; meaningful only if **Cap** is true; PDF 1.7*) A name describing the annotation's caption positioning. Valid values are **Inline**, meaning the caption will be centered inside the line, and **Top**, meaning the caption will be on top of the line. Default value: **Inline**
Measure	dictionary	(*Optional; PDF 1.7*) A measure dictionary (see Table 8.110) that specifies the scale and units that apply to the line annotation.

KEY	TYPE	VALUE
CO	array	(*Optional; meaningful only if* ***Ca*** specifying the offset of the captio is the horizontal offset along the tive value indicating offset to th the left. The second value is the line, with a positive value indicat shift down. Default value: [0, 0] (no offset fro

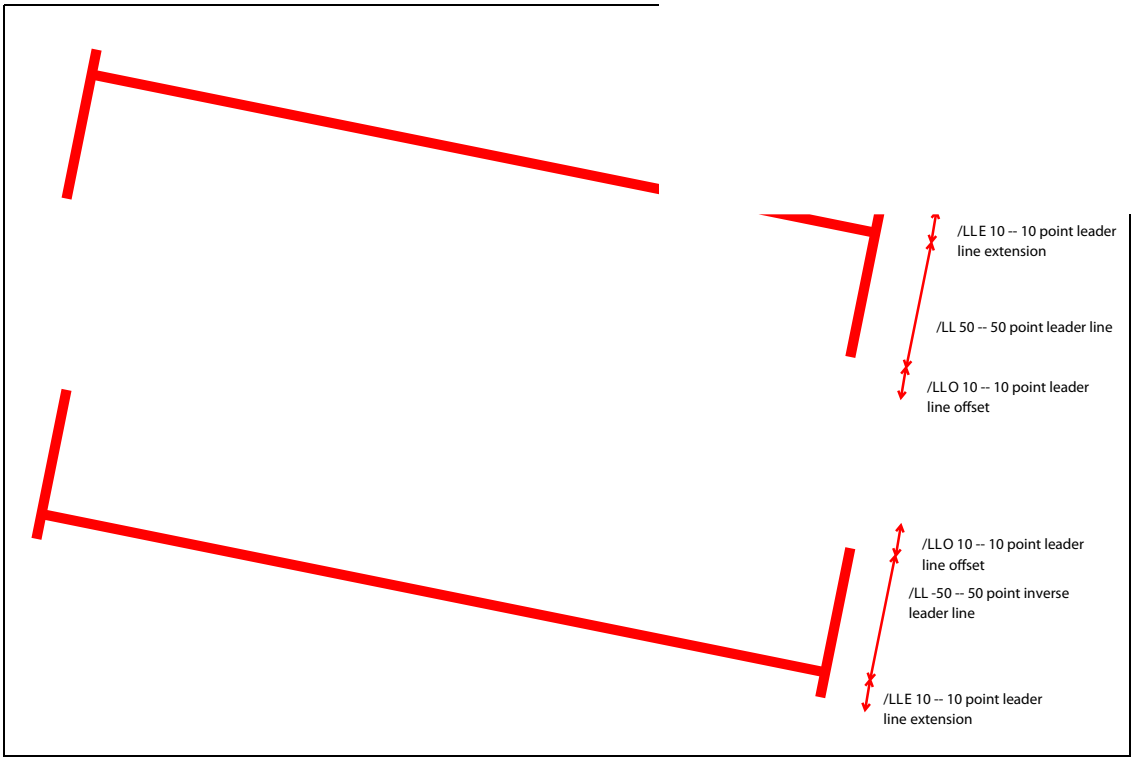


FIGURE 8.5

Figure 8.6 illustrates the effect of including a ca
is specified by setting **Cap** to true.

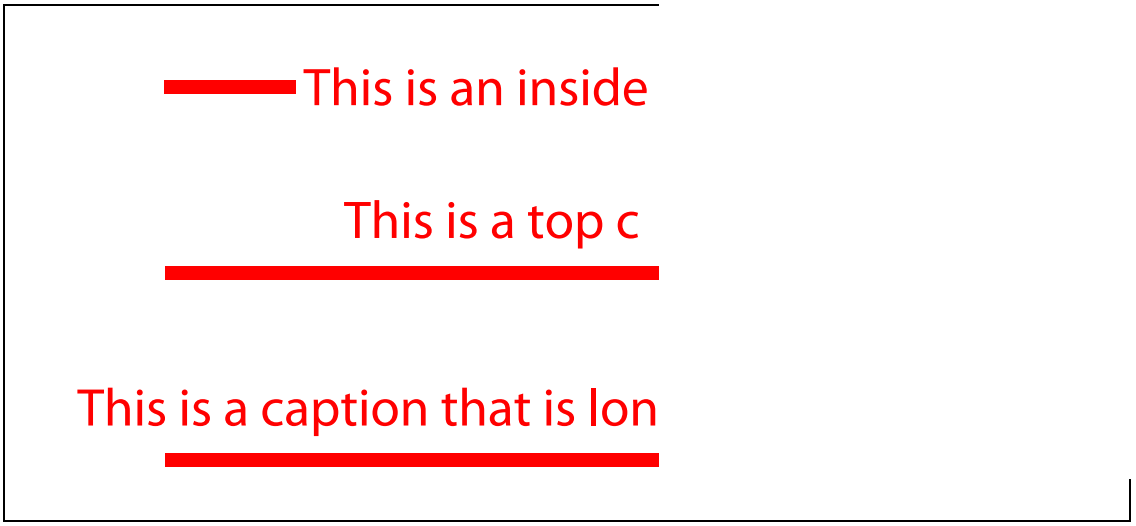


FIGURE 8.6 *Lines with captions appearing as part of the line*

Figure 8.7 illustrates the effect of applying a caption to a line annotation that has a leader offset.

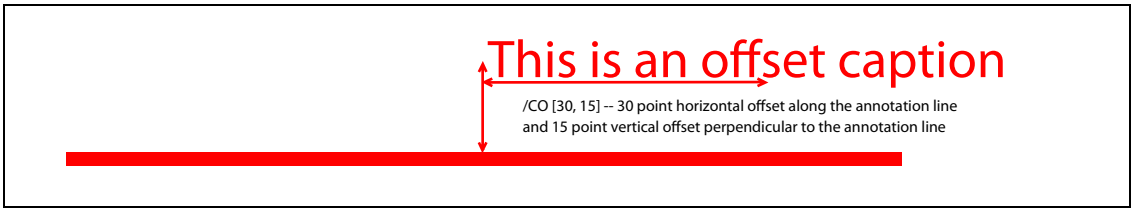


FIGURE 8.7 *Line with a caption appearing as part of the offset*

TABLE 8.27 Line endin

NAME	APPEARANCE	DESCRIPTION
Square		A square filled with the an
Circle		A circle filled with the ann
Diamond		A diamond shape filled wit
OpenArrow		Two short lines meeting in
ClosedArrow		Two short lines meeting in above) and connected by a filled with the annotation's
None		No line ending
Butt		*(PDF 1.5)* A short line at th
ROpenArrow		*(PDF 1.5)* Two short lines i
RClosedArrow		*(PDF 1.5)* A triangular closed arrowhead in the reverse direction from ClosedArrow
Slash		*(PDF 1.6)* A short line at the endpoint approximately 30 degrees clockwise from perpendicular to the line itself

Square and Circle Annotations

Square and *circle annotations (PDF 1.3)* display, respectively, a rectangle or an ellipse on the page. When opened, they display a pop-up window containing the text of the associated note. The rectangle or ellipse is inscribed within the annotation rectangle defined by the annotation dictionary's **Rect** entry (see Table 8.15 on page 606). Figure 8.8 shows two annotations, each with a border width of 18 points. Despite the names *square* and *circle*, the width and height of the annotation rectangle need not be equal. Table 8.28 shows the annotation dictionary en-

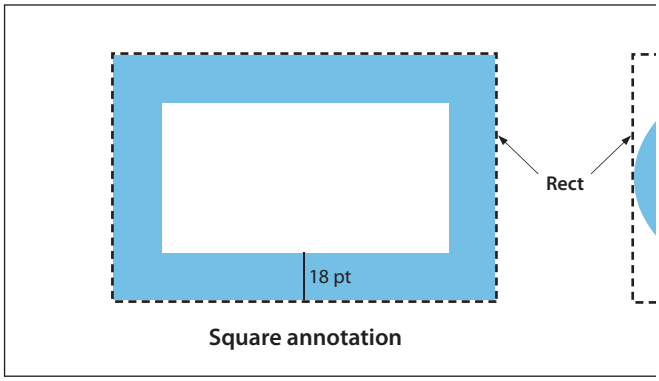


FIGURE 8.8 *Square and circle*

TABLE 8.28 Additional entries specific to a

KEY	TYPE	VALUE
Subtype	name	*(Required)* The type of annotation that this dictionary describes; must be **Square** or **Circle** for a square or circle annotation, respectively.
BS	dictionary	*(Optional)* A border style dictionary (see Table 8.17 on page 611) specifying the line width and dash pattern to be used in drawing the rectangle or ellipse. ***Note:*** *The annotation dictionary's* ***AP*** *entry, if present, takes precedence over the* ***Rect*** *and* ***BS*** *entries; see Table 8.15 on page 606 and Section 8.4.4, "Appearance Streams."*
IC	array	*(Optional; PDF 1.4)* An array of numbers in the range 0.0 to 1.0 specifying the *interior color* with which to fill the annotation's rectangle or ellipse. The number of array elements determines the color space in which the color is defined: 0 No color; transparent 1 **DeviceGray** 3 **DeviceRGB** 4 **DeviceCMYK**
BE	dictionary	*(Optional; PDF 1.5)* A *border effect dictionary* describing an effect applied to the

KEY	TYPE	VALUE
RD	rectangle	*(Optional; PDF 1.5)* A set of four between two rectangles: the **Rect** aries of the underlying square or where a border effect (described yond that of the square or circle. The four numbers correspond t the left, top, right, and bottom co cle, respectively. Each value mus top and bottom differences must the left and right differences mus

Polygon and Polyline Annotations

Polygon annotations (PDF 1.5) display closed p gons may have any number of vertices connecte *tations (PDF 1.5)* are similar to polygons, except not implicitly connected.

TABLE 8.29 Additional entries specific to a polygon or polyline annotation

KEY	TYPE	VALUE
Subtype	name	*(Required)* The type of annotation that this dictionary describes; must be **Polygon** or **PolyLine** for a polygon or polyline annotation, respectively.
Vertices	array	*(Required)* An array of numbers representing the alternating horizontal and vertical coordinates, respectively, of each vertex, in default user space.
LE	array	*(Optional; meaningful only for polyline annotations)* An array of two names specifying the line ending styles. The first and second elements of the array specify the line ending styles for the endpoints defined, respectively, by the first and last pairs of coordinates in the **Vertices** array. Table 8.27 shows the possible values. Default value: [/None /None].
BS	dictionary	*(Optional)* A border style dictionary (see Table 8.17 on page 611) specifying the

KEY	TYPE	VALUE
IC	array	*(Optional; PDF 1.4)* An array of *interior color* with which to fill th number of array elements deter fined: 0 No color; transparent 1 **DeviceGray** 3 **DeviceRGB** 4 **DeviceCMYK**
BE	dictionary	*(Optional; meaningful only for po* scribing an effect applied to the 8.18).
IT	name	*(Optional; PDF 1.6)* A name desc notation (see also Table 8.21). Th **PolygonCloud**, which means t cloud object **PolyLineDimension** (PDF 1.7), which indicates that the polyline annotation is intended to function as a dimension **PolygonDimension** (PDF 1.7), which indicates that the polygon annotation is intended to function as a dimension
Measure	dictionary	(*Optional; PDF 1.7*) A measure dictionary (see Table 8.110) that specifies the scale and units that apply to the annotation.

Text Markup Annotations

Text markup annotations appear as highlights, underlines, strikeouts *(all PDF 1.3)*, or jagged ("squiggly") underlines *(PDF 1.4)* in the text of a document. When opened, they display a pop-up window containing the text of the associated note. Table 8.30 shows the annotation dictionary entries specific to these types of annotations.

TABLE 8.30 Additional entries specific to		
KEY	**TYPE**	**VALUE**
Subtype	name	*(Required)* The type of anno **Highlight**, **Underline**, **Squig** squiggly-underline, or strikeo
QuadPoints	array	*(Required)* An array of $8 \times n$ n laterals in default user spac group of contiguous words in nates for each quadrilateral ar $x_1\ y_1\ x_2\ y_2\ x_3\ y_3\ x_4\ y_4$ specifying the quadrilateral's Figure 8.9). The text is orient (x_1, y_1) and (x_2, y_2). (See impl ***Note:*** *The annotation diction* ***QuadPoints****; see Table 8.15 an*

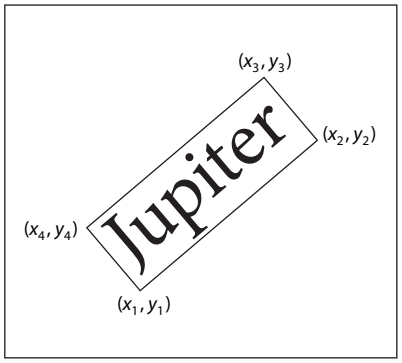


FIGURE 8.9 *QuadPoints specification*

Caret Annotations

TABLE 8.31 Additional entries specific

KEY	TYPE	VALUE
Subtype	name	*(Required)* The type of annotatio for a caret annotation.
RD	rectangle	*(Optional; PDF 1.5)* A set of four between two rectangles: the **Rect** aries of the underlying caret. Su paragraph symbol specified by **S** The four numbers correspond t the left, top, right, and bottom co tively. Each value must be greater tom differences must be less tha right differences must be less tha
Sy	name	*(Optional)* A name specifying a s P A new paragraph sym None No symbol should be associated with the caret. Default value: None.

Rubber Stamp Annotations

A *rubber stamp annotation (PDF 1.3)* displays text or graphics intended to look as if they were stamped on the page with a rubber stamp. When opened, it displays a pop-up window containing the text of the associated note. Table 8.32 shows the annotation dictionary entries specific to this type of annotation.

TABLE 8.32 Additional entries specific to a rubber stamp annotation

KEY	TYPE	VALUE
Subtype	name	*(Required)* The type of annotation that this dictionary describes; must be **Stamp** for a rubber stamp annotation.

KEY	TYPE	VALUE
Name	name	*(Optional)* The name of an icon t applications should provide pred ing standard names: Approved Expe AsIs Expi Confidential Final Departmental ForC Draft ForP Additional names may be suppor ***Note:*** *The annotation dictionary' **Name** entry; see Table 8.15 on pag*

Ink Annotations

An *ink annotation (PDF 1.3)* represents a freeha
more disjoint paths. When opened, it displays a pop-up window containing the text of the associated note. Table 8.33 shows the annotation dictionary entries specific to this type of annotation.

TABLE 8.33 Additional entries specific to an ink annotation

KEY	TYPE	VALUE
Subtype	name	*(Required)* The type of annotation that this dictionary describes; must be **Ink** for an ink annotation.
InkList	array	*(Required)* An array of *n* arrays, each representing a stroked path. Each array is a series of alternating horizontal and vertical coordinates in default user space, specifying points along the path. When drawn, the points are connected by straight lines or curves in an implementation-dependent way. (See implementation note 93 in Appendix H.)
BS	dictionary	*(Optional)* A border style dictionary (see Table 8.17 on page 611) specifying the line width and dash pattern to be used in drawing the paths.

Pop-up Annotations

A *pop-up annotation (PDF 1.3)* displays text in
editing. It typically does not appear alone but is a
tion, its *parent annotation*, and is used for edi
appearance stream or associated actions of its o
entry in the parent's annotation dictionary (see
8.34 shows the annotation dictionary entries spe

TABLE 8.34 Additional entries specific t

KEY	TYPE	VALUE
Subtype	name	*(Required)* The type of anno **Popup** for a pop-up annotatio
Parent	dictionary	*(Optional; must be an indirec* this pop-up annotation is ass ***Note:*** *If this entry is present, entries (see Table 8.15 on page 606) override those of the pop-up annotation itself.*
Open	boolean	*(Optional)* A flag specifying whether the pop-up annotation should initially be displayed open. Default value: **false** (closed).

File Attachment Annotations

A *file attachment annotation (PDF 1.3)* contains a reference to a file, which typically is embedded in the PDF file (see Section 3.10.3, "Embedded File Streams"); see implementation note 95 in Appendix H. For example, a table of data might use a file attachment annotation to link to a spreadsheet file based on that data; activating the annotation extracts the embedded file and gives the user an opportunity to view it or store it in the file system. Table 8.35 shows the annotation dictionary entries specific to this type of annotation.

The **Contents** entry of the annotation dictionary may specify descriptive text re-

TABLE 8.35 Additional entries specific to a

KEY	TYPE	VALUE
Subtype	name	*(Required)* The type of anno **FileAttachment** for a file attac
FS	file specification	*(Required)* The file associated
Name	name	*(Optional)* The name of an i Viewer applications should p the following standard names: Graph Paperclip Additional names may be sup ***Note:*** *The annotation dictiona* ***Name*** *entry; see Table 8.15 Streams."*

Sound Annotations

A *sound annotation (PDF 1.2)* is analogous to a text annotation except that instead of a text note, it contains sound recorded from the computer's microphone or imported from a file. When the annotation is activated, the sound is played. The annotation behaves like a text annotation in most ways, with a different icon (by default, a speaker) to indicate that it represents a sound. Table 8.36 shows the annotation dictionary entries specific to this type of annotation. Sound objects are discussed in Section 9.2, "Sounds."

TABLE 8.36 Additional entries specific to a sound annotation

KEY	TYPE	VALUE
Subtype	name	*(Required)* The type of annotation that this dictionary describes; must be **Sound** for a sound annotation.
S		

KEY	TYPE	VALUE
Name	name	*(Optional)* The name of an icon t applications should provide pre dard names Speaker and Mic. A fault value: Speaker. ***Note:*** *The annotation dictionary' **Name** entry; see Table 8.15 on pag*

Movie Annotations

A *movie annotation (PDF 1.2)* contains animate sented on the computer screen and through the s activated, the movie is played. Table 8.37 shows t specific to this type of annotation. Movies are di

TABLE 8.37 Additional entries specific

KEY	TYPE	VALUE
Subtype	name	*(Required)* The type of annotation that this dictionary describes; must be **Movie** for a movie annotation.
T	text string	(*Optional*) The title of the movie annotation. Movie actions (page 664) can use this title to reference the movie annotation.
Movie	dictionary	*(Required)* A movie dictionary describing the movie's static characteristics (see Section 9.3, "Movies").
A	boolean or dictionary	*(Optional)* A flag or dictionary specifying whether and how to play the movie when the annotation is activated. If this value is a dictionary, it is a movie activation dictionary (see Section 9.3, "Movies") specifying how to play the movie. If the value is the boolean **true**, the movie should be played using default activation parameters. If the value is **false**, the movie should not be played. Default value: **true**.

Screen Annotations

TABLE 8.38 Additional entries specific

KEY	TYPE	VALUE
Subtype	name	*(Required)* The type of annotatio for a screen annotation.
T	text string	(*Optional*) The title of the screen
MK	dictionary	*(Optional)* An appearance charac of this dictionary provides the ic to by the screen annotation's **AP** e
A	dictionary	*(Optional; PDF 1.1)* An action to (see Section 8.5, "Actions").
AA	dictionary	*(Optional; PDF 1.2)* An addition tation's behavior in response to v Events").

In addition to the above entries, screen annotations use the common entries in the annotation dictionary (see Table 8.15) in the following ways:

- The **P** entry is required for a screen annotation referenced by a rendition action. It must reference a valid page object, and the annotation must be present in the page's **Annots** array for the action to be valid.
- The **AP** entry refers to an appearance dictionary (see Table 8.19) whose normal appearance provides the visual appearance for a screen annotation that is used for printing and default display when a media clip is not being played. If **AP** is not present, the screen annotation has no default visual appearance and is not printed.

Widget Annotations

Interactive forms (see Section 8.6, "Interactive Forms") use *widget annotations (PDF 1.2)* to represent the appearance of fields and to manage user interactions.

annotation dictionary entries specific to this typ
and fields are discussed at length in Section 8.6.

TABLE 8.39 Additional entries specific

KEY	TYPE	VALUE
Subtype	name	*(Required)* The type of annotatio for a widget annotation.
H	name	*(Optional)* The annotation's *highl* the mouse button is pressed or h N (None) No highlighting. I (Invert) Invert the conte O (Outline) Invert the ann P (Push) Display the ann 8.4.4, "Appearance Strea the contents of the ann pushed below the surface of the page. T (Toggle) Same as P (which is preferred). A highlighting mode other than P overrides any down appearance defined for the annotation. Default value: I.
MK	dictionary	*(Optional)* An appearance characteristics dictionary (see Table 8.40) to be used in constructing a dynamic appearance stream specifying the annotation's visual presentation on the page. *The name **MK** for this entry is of historical significance only and has no direct meaning.*
A	dictionary	*(Optional; PDF 1.1)* An action to be performed when the annotation is activated (see Section 8.5, "Actions").
AA	dictionary	*(Optional; PDF 1.2)* An additional-actions dictionary defining the annotation's behavior in response to various trigger events (see Section 8.5.2, "Trigger Events").
B		

The **MK** entry can be used to provide an *appeara* taining additional information for constructi stream. Table 8.40 shows the contents of this dic

TABLE 8.40 Entries in an appearance ch

KEY	TYPE	VALUE
R	integer	*(Optional)* The number of degr counterclockwise relative to th Default value: 0.
BC	array	*(Optional)* An array of numbers i widget annotation's border. The space in which the color is define 0 No color; transparent 1 **DeviceGray** 3 **DeviceRGB** 4 **DeviceCMYK**
BG	array	*(Optional)* An array of numbers in the range 0.0 to 1.0 specifying the color of the widget annotation's background. The number of array elements determines the color space, as described above for **BC**.
CA	text string	*(Optional; button fields only)* The widget annotation's *normal caption*, displayed when it is not interacting with the user. ***Note:*** *Unlike the remaining entries listed below, which apply only to widget annotations associated with pushbutton fields (see "Pushbuttons" on page 686), the* ***CA*** *entry can be used with any type of button field, including check boxes ("Check Boxes" on page 686) and radio buttons ("Radio Buttons" on page 688).*
RC	text string	*(Optional; pushbutton fields only)* The widget annotation's *rollover caption*, displayed when the user rolls the cursor into its active area without pressing the mouse button.
AC	text string	*(Optional; pushbutton fields only)* The widget annotation's *alternate (down) caption*, displayed when the mouse button is pressed within its active area.
I		

KEY	TYPE	VALUE
RI	stream	*(Optional; pushbutton fields only;* defining the widget annotation's cursor into its active area withou
IX	stream	*(Optional; pushbutton fields only;* defining the widget annotation mouse button is pressed within it
IF	dictionary	*(Optional; pushbutton fields only)* 719) specifying how to display th rectangle. If present, the icon fit (normal, rollover, and alternate).
TP	integer	*(Optional; pushbutton fields only)* the widget annotation's caption r 0 No icon; caption only 1 No caption; icon only 2 Caption below the icon 3 Caption above the icon 4 Caption to the right of the icon 5 Caption to the left of the icon 6 Caption overlaid directly on the icon Default value: 0.

Printer's Mark Annotations

A *printer's mark annotation (PDF 1.4)* represents a graphic symbol, such as a registration target, color bar, or cut mark, added to a page to assist production personnel in identifying components of a multiple-plate job and maintaining consistent output during production. See Section 10.10.2, "Printer's Marks," for further discussion.

Trap Network Annotations

trap network annotation, whose **Subtype** entry is always the last element in the page object's **An** page 144). See Section 10.10.5, "Trapping Suppo

Watermark Annotations

A *watermark annotation (PDF 1.6)* is used to rep to be printed at a fixed size and position on a p of the printed page. The **FixedPrint** entry of a (see Table 8.41) is a dictionary that contains valu sition of the annotation (see Table 8.42).

Watermark annotations have no pop-up windo When displaying a watermark annotation on-sc use the dimensions of the media box as the page behavior is the same as for other annotations.

Note: *Since many printing devices have nonprint that such margins be taken into consideration when positioning watermark annotations near the edge of a page.*

TABLE 8.41 Additional entries specific to a watermark annotation

KEY	TYPE	VALUE
Subtype	name	*(Required)* The type of annotation that this dictionary describes; must be **Watermark** for a watermark annotation.
FixedPrint	dictionary	*(Optional)* A *fixed print dictionary* (see Table 8.42) that specifies how this annotation should be drawn relative to the dimensions of the target media. If this entry is not present, the annotation is drawn without any special consideration for the dimensions of the target media. ***Note:*** *If the dimensions of the target media are not known at the time of drawing, drawing is done relative to the dimensions specified by the page's* ***MediaBox*** *entry (see Table 3.27).*

TABLE 8.42 Entries in a fixed p

KEY	TYPE	VALUE
Type	name	*(Required)* Must be **FixedPrint**.
Matrix	array	*(Optional)* The matrix used to tr dering. Default value: the identity matrix ***Note:*** *When positioning content this entry be used to provide a rea*
H	number	*(Optional)* The amount to transl centage of the width of the targe **MediaBox**). 1.0 represents 100% recommended, since they may ca Default value: 0.
V	number	*(Optional)* The amount to transl centage of the height of the target media (or if unknown, the height of the page's **MediaBox**). 1.0 represents 100% and 0.0 represents 0%. Negative values are not recommended, since they may cause content to be drawn off the page. Default value: 0.

When rendering a watermark annotation with a **FixedPrint** entry, the following behavior occurs:

- The annotation's rectangle (as specified by its **Rect** entry) is translated to the origin and transformed by the **Matrix** entry of its **FixedPrint** dictionary to produce a quadrilateral with arbitrary orientation.
- The *transformed annotation rectangle* is defined as the smallest upright rectangle that encompasses this quadrilateral; it is used in place of the annotation rectangle referred to in steps 2 and 3 of Algorithm 8.1 on page 612.

In addition, given a matrix B that maps a scaled and rotated page into the default

Example 8.10

```
8 0 obj                          % Watermark appe
   <<
      /Length ...
      /Subtype /Form
      /Resources ...
      /BBox ...
   >>
   stream
   ...
      BT
      /F1 1 Tf
      36 0 0 36 0 -36 Tm
      (Do Not Build) Tx
      ET
      ...
   endstream
endobj

9 0 obj                          % Watermark annotation
   <<
      /Rect ...
      /Type /Annot
      /Subtype /Watermark
      /FixedPrint 10 0 R
      /AP <</N 8 0 R>>
   >>

% in the page dictionary
   /Annots [9 0 R]

10 0 obj                         % Fixed print dictionary
   <<
      /Type /FixedPrint
      /Matrix [1 0 0 1 72 -72]   % Translate one inch right and one inch down
      /H 0
      /V 1.0                     % Translate the full height of the page vertically
```

In situations other than the usual case where the page size, watermark annotations with a **FixedPri** following manner:

- When page tiling is selected in a viewer appli is printed on multiple pages), the annotations and position on each page to ensure that any legible on each printed page.
- When *n*-up printing is selected (that is, mult single page), the annotations are printed at the as if the dimensions of the printed page were page. This ensures that any enclosed content other pages, thus rendering it illegible. (See i dix H.)

8.5 Actions

Instead of simply jumping to a destination in the document, an annotation or outline item can specify an *action (PDF 1.1)* for the viewer application to perform, such as launching an application, playing a sound, or changing an annotation's appearance state. The optional **A** entry in the annotation or outline item dictionary (see Tables 8.15 on page 606 and 8.4 on page 585) specifies an action to be performed when the annotation or outline item is activated; in PDF 1.2, a variety of other circumstances may trigger an action as well (see Section 8.5.2, "Trigger Events"). In addition, the optional **OpenAction** entry in a document's catalog (Section 3.6.1, "Document Catalog") may specify an action to be performed when the document is opened. PDF includes a wide variety of standard action types, described in detail in Section 8.5.3, "Action Types."

8.5.1 Action Dictionaries

An *action dictionary* defines the characteristics and behavior of an action. Table 8.43 shows the required and optional entries that are common to all action

TABLE 8.43 Entries common to all

KEY	TYPE	VALUE
Type	name	*(Optional)* The type of present, must be **Action** fo
S	name	*(Required)* The type of acti on page 653 for specific va
Next	dictionary or array	*(Optional; PDF 1.2)* The formed after the action re a single action dictionary formed in order; see below

The action dictionary's **Next** entry *(PDF 1.2)* al chained together. For example, the effect of cli mouse might be to play a sound, jump to a new that the **Next** entry is not restricted to a single ac actions, each of which in turn may have a **Next** e thus form a tree instead of a simple linked list. Actions within each **Next** array are executed in order, each followed in turn by any actions specified in *its* **Next** entry, and so on recursively. Viewer applications should attempt to provide reasonable behavior in anomalous situations. For example, self-referential actions should not be executed more than once, and actions that close the document or otherwise render the next action impossible should terminate the execution sequence. Applications should also provide some mechanism for the user to interrupt and manually terminate a sequence of actions.

PDF 1.5 introduces transition actions, which allow the control of drawing during a sequence of actions; see "Transition Actions" on page 670.

Note: *No action should modify its own action dictionary or any other in the action tree in which it resides. The effect of such modification on subsequent execution of actions in the tree is undefined.*

tion. In PDF 1.4, the document catalog dictiona Catalog") may also contain an **AA** entry for trigg as a whole. Tables 8.44 to 8.47 show the conten implementation notes 98 and 99 in Appendix H.

PDF 1.5 introduces four trigger events to suppor

- The **PO** and **PC** entries have a similar function object's additional-actions dictionary (see Ta these triggers with annotations allows annota and greatly simplifies authoring. For example, tions can be copied or moved between pages w actions to be changed.
- The **PV** and **PI** entries allow a distinction betwe that are visible. At any one time, only a singl viewer application, while more than one page page layout.

Note: *For these trigger events, the values of the flags specified by the annotation's* ***F*** *entry (see Section 8.4.2, "Annotation Flags") have no bearing on whether a given trigger event occurs.*

TABLE 8.44 Entries in an annotation's additional-actions dictionary

KEY	TYPE	VALUE
E	dictionary	*(Optional; PDF 1.2)* An action to be performed when the cursor enters the annotation's active area.
X	dictionary	*(Optional; PDF 1.2)* An action to be performed when the cursor exits the annotation's active area.
D	dictionary	*(Optional; PDF 1.2)* An action to be performed when the mouse button is pressed inside the annotation's active area. (The name **D** stands for "down.")
U	dictionary	*(Optional; PDF 1.2)* An action to be performed when the mouse button is released inside the annotation's active area. (The name **U** stands for "up.")
F		

KEY	TYPE	VALUE
BI	dictionary	*(Optional; PDF 1.2; widget annotations* be performed when the annotation lo "blurred.")
PO	dictionary	*(Optional; PDF 1.5)* An action to be pe tion is opened (for example, when the page or by means of a link annotation o **O** action in the page's additional-a **OpenAction** entry in the document cat
PC	dictionary	*(Optional; PDF 1.5)* An action to be pe tion is closed (for example, when the us lows a link annotation or outline item) the page's additional-actions dictionary
PV	dictionary	*(Optional; PDF 1.5)* An action to be pe tion becomes visible in the viewer appli
PI	dictionary	*(Optional; PDF 1.5)* An action to be performed when the page containing the annotation is no longer visible in the viewer application's user interface.

TABLE 8.45 Entries in a page object's additional-actions dictionary

KEY	TYPE	VALUE
O	dictionary	*(Optional; PDF 1.2)* An action to be performed when the page is opened (for example, when the user navigates to it from the next or previous page or by means of a link annotation or outline item). This action is independent of any that may be defined by the **OpenAction** entry in the document catalog (see Section 3.6.1, "Document Catalog") and is executed after such an action. (See implementation note 100 in Appendix H.)
C	dictionary	*(Optional; PDF 1.2)* An action to be performed when the page is closed (for example, when the user navigates to the next or previous page or follows a link annotation or an outline item). This action applies to the page being closed and is executed before any other page is opened. (See implementation note 100 in Appendix H.)

TABLE 8.46 Entries in a form field's addit

KEY	TYPE	VALUE
K	dictionary	*(Optional; PDF 1.3)* A JavaScript actio stroke into a text field or combo box o This action can check the keystroke for
F	dictionary	*(Optional; PDF 1.3)* A JavaScript action display its current value. This action ca
V	dictionary	*(Optional; PDF 1.3)* A JavaScript acti changed. This action can check the new idate.")
C	dictionary	*(Optional; PDF 1.3)* A JavaScript action field when that of another field changes in which the document's fields are reca active form dictionary (see Section 8.6.

TABLE 8.47 Entries in the document catalog's additional-actions dictionary

KEY	TYPE	VALUE
WC	dictionary	*(Optional; PDF 1.4)* A JavaScript action to be performed before closing a document. (The name **WC** stands for "will close.")
WS	dictionary	*(Optional; PDF 1.4)* A JavaScript action to be performed before saving a document. (The name **WS** stands for "will save.")
DS	dictionary	*(Optional; PDF 1.4)* A JavaScript action to be performed after saving a document. (The name **DS** stands for "did save.")
WP	dictionary	*(Optional; PDF 1.4)* A JavaScript action to be performed before printing a document. (The name **WP** stands for "will print.")
DP	dictionary	*(Optional; PDF 1.4)* A JavaScript action to be performed after printing a document. (The name **DP** stands for "did print.")

- A notion of *location*—that is, an indication of pointing. Location is typically denoted by a sc
- A notion of *focus*—that is, which element in th ing with the user. In many systems, this eleme a focus rectangle, or a color change.

PDF viewer applications must ensure the presen sponding actions to be executed correctly. Mou ject to the following constraints:

- An **E** (enter) event can occur only when the m
- An **X** (exit) event cannot occur without a prec
- A **U** (up) event cannot occur without a precedi
- In the case of overlapping or nested annotatio active area causes an **X** event to occur for the fi

Note: *The field-related trigger events **K** (keystroke), **F** (format), **V** (validate), and **C** (calculate) are not defined for button fields (see "Button Fields" on page 685). The effects of an action triggered by one of these events are limited only by the action itself and can occur outside the described scope of the event. For example, even though the **F** event is used to trigger actions that format field values prior to display, it is possible for an action triggered by this event to perform a calculation or make any other modification to the document.*

*These field-related trigger events can occur either through user interaction or programmatically, such as in response to the **NeedAppearances** entry in the interactive form dictionary (see Section 8.6.1, "Interactive Form Dictionary"), importation of FDF data (Section 8.6.6, "Forms Data Format"), or JavaScript actions ("JavaScript Actions" on page 709). For example, the user's modifying a field value can trigger a cascade of calculations and further formatting and validation for other fields in the document.*

8.5.3 Action Types

TABLE 8.48 Action t

ACTION TYPE	DESCRIPTION	
GoTo	Go to a destination in the current document.	
GoToR	("Go-to remote") Go to a destination in another document.	
GoToE	("Go-to embedded"; *PDF 1.6*) Go to a destinatio embedded file.	
Launch	Launch an application, usually to open a file.	
Thread	Begin reading an article thread.	
URI	Resolve a uniform resource identifier.	
Sound	*(PDF 1.2)* Play a sound.	
Movie	*(PDF 1.2)* Play a movie.	
Hide	*(PDF 1.2)* Set an annotation's Hidden flag.	"Hide Actions" on page 665
Named	*(PDF 1.2)* Execute an action predefined by the viewer application.	"Named Actions" on page 666
SubmitForm	*(PDF 1.2)* Send data to a uniform resource locator.	"Submit-Form Actions" on page 703
ResetForm	*(PDF 1.2)* Set fields to their default values.	"Reset-Form Actions" on page 707
ImportData	*(PDF 1.2)* Import field values from a file.	"Import-Data Actions" on page 708
JavaScript	*(PDF 1.3)* Execute a JavaScript script.	"JavaScript Actions" on page 709
SetOCGState	*(PDF 1.5)* Set the states of optional content groups.	"Set-OCG-State Actions" on page 667
Rendition	*(PDF 1.5)* Controls the playing of multimedia content.	"Rendition Actions" on page 668
Trans	*(PDF 1.5)* Updates the display of a document, using a transition dictionary.	"Transition Actions" on page 670
GoTo3DView	*(PDF 1.6)* Set the current view of a 3D annotation	"Go-To-3D-View Actions" on page

Go-To Actions

A *go-to action* changes the view to a specified magnification factor). Table 8.49 shows the acti this type of action.

TABLE 8.49 Additional entries speci

KEY	TYPE	VALUE
S	name	*(Required)* The type of action that go-to action.
D	name, byte string, or array	*(Required)* The destination to jump

Specifying a go-to action in the **A** entry of a lin Tables 8.24 on page 622 and 8.4 on page 585) has destination directly with the **Dest** entry. For example, the link annotation shown in Example 8.11, which uses a go-to action, has the same effect as the one in Example 8.9 on page 623, which specifies the destination directly. However, the go-to action is less compact and is not compatible with PDF 1.0; therefore, using a direct destination is preferable.

Example 8.11

```
93 0 obj
   << /Type /Annot
      /Subtype /Link
      /Rect [71 717 190 734]
      /Border [16 16 1]
      /A << /Type /Action
            /S /GoTo
            /D [3 0 R /FitR –4 399 199 533]
         >>
   >>
```

Remote Go-To Actions

A *remote go-to action* is similar to an ordinary g nation in another PDF file instead of the current dictionary entries specific to this type of action.

Note: *Remote go-to actions cannot be used with e To Actions" on page 655".*

TABLE 8.50 Additional entries specific t

KEY	TYPE	VALUE
S	name	*(Required)* The type of action for a remote go-to action.
F	file specification	*(Required)* The file in which t
D	name, byte string, or array	*(Required)* The destination to the value is an array definin "Explicit Destinations" on page 582), its first element must be a page number within the remote document rather than an indirect reference to a page object in the current document. The first page is numbered 0.
NewWindow	boolean	*(Optional; PDF 1.2)* A flag specifying whether to open the destination document in a new window. If this flag is **false**, the destination document replaces the current document in the same window. If this entry is absent, the viewer application should behave in accordance with the current user preference.

Embedded Go-To Actions

An *embedded go-to action (PDF 1.6)* is similar to a remote go-to action but allows jumping to or from a PDF file that is embedded in another PDF file (see "Embedded File Streams" on page 184). Embedded files may be associated with file attachment annotations (see "File Attachment Annotations" on page 637) or with entries in the **EmbeddedFiles** name tree (see Section 3.6.3, "Name Dictionary"). Embedded files may in turn contain embedded files. Table 8.51 shows the action

Embedded go-to actions provide a complete faci
hierarchy of nested embedded files and another
archy. The following terminology is used:

- The *source* is the document containing the em
- The *target* is the document in which the destin

  The **T** entry in the action dictionary is a target
  in relation to the source, in much the same
  the physical relationship between two files in
  may be nested recursively to specify one or
  reaching the final one. As the hierarchy is navi
  referred to as the *current document*. Initially,
  ment.

  Note: *It is an error for a target dictionary to h
  one where a target dictionary refers to itself). V
  to detect such cases and refuse to execute the ac*

- A *child* document is one that is embedded within another PDF file.
- The document in which a file is embedded is its *parent*.
- A *root document* is one that is not embedded in another PDF file. The target and source may be contained in root documents or embedded documents.

TABLE 8.51 Additional entries specific to an embedded go-to action

KEY	TYPE	VALUE
S	name	*(Required)* The type of action that this dictionary describes; must be **GoToE** for an embedded go-to action.
F	file specification	*(Optional)* The root document of the target relative to the root document of the source. If this entry is absent, the source and target share the same root document.
D	name, byte string,	*(Required)* The destination in the target to jump to (see Section 8.2.1, "Destinations").
N		

KEY	TYPE	VALUE
T	dictionary	*(Optional if **F** is present; otherw* specifying path information t specifies one element in the fu dictionaries specifying additio

TABLE 8.52 Entries specific to a t

KEY	TYPE	VALUE
R	name	*(Required)* Specifies the relati target (which may be an inter the parent of the current docu document).
N	byte string	*(Required if the value of **R** is name tree; otherwise, it mu* **EmbeddedFiles** name tree.
P	integer or byte string	*(Required if the value of **R** is **C** and the target is associated with a file attachment annotation; otherwise, it must be absent)* If the value is an integer, it specifies the page number (zero-based) in the current document containing the file attachment annotation. If the value is a string, it specifies a named destination in the current document that provides the page number of the file attachment annotation.
A	integer or text string	*(Required if the value of **R** is **C** and the target is associated with a file attachment annotation; otherwise, it must be absent)* If the value is an integer, it specifies the index (zero-based) of the annotation in the **Annots** array (see Table 3.27) of the page specified by **P**. If the value is a text string, it specifies the value of **NM** in the annotation dictionary (see Table 8.15).
T	dictionary	*(Optional)* A target dictionary specifying additional path information to the target document. If this entry is absent, the current document is the target file containing the destination.

Example 8.12 illustrates several possible relation
Each object shown is an action dictionary for an

Example 8.12

```
1 0 obj                    % Link to a child
   << /Type /Action
      /S /GoToE
      /D (Chapter 1)
      /T  << /R /C
             /N (Embedded document) >>
   >>
endobj


2 0 obj                    % Link to the parent
   << /Type /Action
      /S /GoToE
      /D (Chapter 1)
      /T << /R /P >>
   >>
endobj

3 0 obj                    % Link to a sibling
   << /Type /Action
      /S /GoToE
      /D (Chapter 1)
      /T <<  /R /P
             /T <<  /R /C
                    /N (Another embedded document) >>
          >>
   >>
endobj

4 0 obj                    % Link to an embedded file in an external document
   << /Type /Action
      /S /GoToE
      /D (Chapter 1)
```

```
5 0 obj                     % Link from an embedded
   << /Type /Action
      /S /GoToE
      /D (Chapter 1)
      /F (someFile.pdf)
   >>
endobj

6 0 obj                     % Link to a grandchild
   << /Type /Action
      /S /GoToE
      /D (Chapter 1)
      /T <<  /R  /C
               /N (Embedded document)
               /T << /R /C
                        /P (A destination name)
                        /A (annotName)
                     >>
            >>
   >>
endobj

7 0 obj                     % Link to a niece/nephew through the source's parent
   << /Type /Action
      /S /GoToE
      /D (destination)
      /T <<  /R /P
               /T << /R /C
                        /N (Embedded document)
                        /T <<  /R /C
                                 /P 3
                                 /A (annotName)
                              >>
                     >>
            >>
   >>
endobj
```

The optional **Win**, **Mac**, and **Unix** entries allow platform-specific parameters for launching th such entry is present for the given platform, t 8.54 shows the platform-specific launch param Parameters for the Mac OS and UNIX platform of publication.

TABLE 8.53 Additional entries specif

KEY	TYPE	VALUE
S	name	*(Required)* The type of action for a launch action.
F	file specification	*(Required if none of the entries* be launched or the document and the viewer application do it should do nothing.
Win	dictionary	*(Optional)* A dictionary conta Table 8.54; see also implementation note 101 in Appendix H).
Mac	(undefined)	*(Optional)* Mac OS–specific launch parameters; not yet defined.
Unix	(undefined)	*(Optional)* UNIX-specific launch parameters; not yet defined.
NewWindow	boolean	*(Optional; PDF 1.2)* A flag specifying whether to open the destination document in a new window. If this flag is **false**, the destination document replaces the current document in the same window. If this entry is absent, the viewer application should behave in accordance with the current user preference. This entry is ignored if the file designated by the **F** entry is not a PDF document.

TABLE 8.54 Entries in a Windows launch parameter dictionary

KEY	TYPE	VALUE
F	byte string	*(Required)* The file name of the application to be launched or the document

KEY	TYPE	VALUE
D	byte string	*(Optional)* A bye string specif tax.
O	ASCII string	*(Optional)* An ASCII string s open Open a documen print Print a documen If the **F** entry designates an ap nored and the application is l
P	byte string	*(Optional)* A parameter strin the **F** entry. This entry should

Thread Actions

A *thread action* jumps to a specified bead on an "Articles"), in either the current document or a the action dictionary entries specific to this type of action.

TABLE 8.55 Additional entries specific to a thread action

KEY	TYPE	VALUE
S	name	*(Required)* The type of action that this dictionary describes; must be **Thread** for a thread action.
F	file specification	*(Optional)* The file containing the thread. If this entry is absent, the thread is in the current file.
D	dictionary, integer, or text string	*(Required)* The destination thread, specified in one of the following forms: • An indirect reference to a thread dictionary (see Section 8.3.2, "Articles"). In this case, the thread must be in the current file. • The index of the thread within the **Threads** array of its document's catalog (see Section 3.6.1, "Document Catalog"). The first thread in the array has index 0.

KEY	TYPE	VALUE
B	dictionary or integer	*(Optional)* The bead in the de ing forms: • An indirect reference to a b this case, the thread must b • The index of the bead wit index 0.

URI Actions

A *uniform resource identifier* (URI) is a string source on the Internet—typically a file that is th although it can also resolve to a query or other ternet RFC 2396, *Uniform Resource Identifiers (* liography.)

A *URI action* causes a URI to be resolved. Table 8.56 shows the action dictionary entries specific to this type of action. (See implementation notes 102 and 103 in Appendix H.)

TABLE 8.56 Additional entries specific to a URI action

KEY	TYPE	VALUE
S	name	*(Required)* The type of action that this dictionary describes; must be **URI** for a URI action.
URI	ASCII string	*(Required)* The uniform resource identifier to resolve, encoded in 7-bit ASCII.
IsMap	boolean	*(Optional)* A flag specifying whether to track the mouse position when the URI is resolved (see below). Default value: **false**. This entry applies only to actions triggered by the user's clicking an annotation; it is ignored for actions associated with outline items or with a document's **OpenAction** entry.

the **Rect** entry in the annotation with which the
ample, if the mouse coordinates in user space
rectangle extends from (ll_x, ll_y) at the lower-left
the final coordinates (x_f, y_f) are as follows:

$$(x_f = x_m - ll_x)$$

$$y_f = ur_y - y_m$$

If the resulting coordinates (x_f, y_f) are fractiona
nearest integer values. They are then appended t
ed by commas and preceded by a question mark,

```
http://www.adobe.com/intro?100,200
```

To support URI actions, a PDF document's catal
Catalog") may include a **URI** entry whose value i
publication, only one entry is defined for such a

TABLE 8.57 Entry in a URI dictionary

KEY	TYPE	VALUE
Base	ASCII string	*(Optional)* The *base URI* to be used in resolving relative URI references. URI actions within the document may specify URIs in partial form, to be interpreted relative to this base address. If no base URI is specified, such partial URIs are interpreted relative to the location of the document itself. The use of this entry is parallel to that of the body element <BASE>, as described in the *HTML 4.01 Specification* (see the Bibliography).

The **Base** entry allows the URI of the document to be recorded in situations in which the document may be accessed out of context. For example, if a document has been moved to a new location but contains relative links to other documents that have not been moved, the **Base** entry could be used to refer such links to the true location of the other documents, rather than that of the moved document.

TABLE 8.58 Additional entries specif

KEY	TYPE	VALUE
S	name	*(Required)* The type of action for a sound action.
Sound	stream	*(Required)* A sound object de "Sounds"; see also implement
Volume	number	*(Optional)* The volume at wh see implementation note 106 i
Synchronous	boolean	*(Optional)* A flag specifying asynchronously; see impleme **true**, the viewer application r tion other than canceling th played. Default value: **false**.
Repeat	boolean	*(Optional)* A flag specifying entry is present, the **Synchron**
Mix	boolean	*(Optional)* A flag specifying whether to mix this sound with any other sound already playing; see implementation note 107 in Appendix H. If this flag is **false**, any previously playing sound is stopped before starting this sound; this can be used to stop a repeating sound (see **Repeat**, above). Default value: **false**.

Movie Actions

A *movie action (PDF 1.2)* can be used to play a movie in a floating window or within the annotation rectangle of a movie annotation (see "Movie Annotations" on page 639 and Section 9.3, "Movies"). The movie annotation must be associated with the page that is the destination of the link annotation or outline item containing the movie action, or with the page object with which the action is associated. (See implementation note 108 in Appendix H.)

Note: *A movie action by itself does not guarantee that the page the movie is on will*

annotation provide the default values. Any infor
tion dictionary overrides these values.

TABLE 8.59 Additional entries specif

KEY	TYPE	VALUE
S	name	*(Required)* The type of action that movie action.
Annotation	dictionary	*(Optional)* An indirect reference to played.
T	text string	*(Optional)* The title of a movie an ***Note:*** *The dictionary must include*
Operation	name	*(Optional)* The operation to be per Play — Start playing the tionary's **Mode** en currently paused, ing (or to the starting point specified by the dictionary's **Start** entry, if present). Stop — Stop playing the movie. Pause — Pause a playing movie. Resume — Resume a paused movie. Default value: Play.

Hide Actions

A *hide action (PDF 1.2)* hides or shows one or more annotations on the screen by setting or clearing their Hidden flags (see Section 8.4.2, "Annotation Flags"). This type of action can be used in combination with appearance streams and trigger events (Sections 8.4.4, "Appearance Streams," and 8.5.2, "Trigger Events") to display pop-up help information on the screen. For example, the **E** (enter) and **X** (exit) trigger events in an annotation's additional-actions dictionary can be used to

TABLE 8.60 Additional entries spec

KEY	TYPE	VALUE
S	name	*(Required)* The type of action that th action.
T	dictionary, text string, or array	*(Required)* The annotation or annot of the following forms: • An indirect reference to an annot • A text string giving the fully qu whose associated widget annotati Names" on page 676) • An array of such dictionaries or t
H	boolean	*(Optional)* A flag indicating whether Default value: **true**.

Named Actions

Table 8.61 lists several *named actions (PDF 1.2)* that PDF viewer applications are expected to support; further names may be added in the future. (See implementation notes 111 and 112 in Appendix H.)

TABLE 8.61 Named actions

NAME	ACTION
NextPage	Go to the next page of the document.
PrevPage	Go to the previous page of the document.
FirstPage	Go to the first page of the document.
LastPage	Go to the last page of the document.

Note: *Viewer applications may support additional, nonstandard named actions, but*

TABLE 8.62 Additional entries specif

KEY	TYPE	VALUE
S	name	*(Required)* The type of action that this di action.
N	name	*(Required)* The name of the action to be

Set-OCG-State Actions

A *set-OCG-state action (PDF 1.5)* sets the state groups (see Section 4.10, "Optional Content"). T nary entries specific to this type of action.

TABLE 8.63 Additional entries specific t

KEY	TYPE	VALUE
S	name	*(Required)* The type of action that this dictionary describes; must be **SetOCGState** for a set-OCG-state action.
State	array	*(Required)* An array consisting of any number of sequences beginning with a name object (**ON**, **OFF**, or **Toggle**) followed by one or more optional content group dictionaries. The array elements are processed from left to right; each name is applied to the subsequent groups until the next name is encountered: • **ON** sets the state of subsequent groups to **ON** • **OFF** sets the state of subsequent groups to **OFF** • **Toggle** reverses the state of subsequent groups.
PreserveRB	boolean	*(Optional)* If **true**, indicates that radio-button state relationships between optional content groups (as specified by the **RBGroups** entry in the current configuration dictionary; see Table 4.51 on page 376) should be preserved when the states in the **State** array are applied. That is, if a group is set to **ON** (either by **ON** or **Toggle**) during processing of the **State** array, any other groups belonging to the same radio-button group are turned **OFF**. If a group is set to **OFF**, there is no effect on other groups.

When a set-OCG-state action is performed, the
to right. Each name is applied to subsequent gr
name is encountered, as shown in the following

Example 8.13

```
<< /S /SetOCGState
   /State [/OFF 2 0 R 3 0 R /Toggle 16 0 R 19 0 R /ON
>>
```

A group can appear more than once in the **State**
is encountered, based on the most recent name
tained [/OFF 1 0 R /Toggle 1 0 R], the group's state
performed. **ON**, **OFF** and **Toggle** sequences hav
one sequence in the array may contain the same

Note: *While the specification allows a group to a array, this is not intended to implement animatio operations. PDF processing applications are free to apply only the net changes simultaneously to all affected groups before redrawing.*

Rendition Actions

A *rendition action (PDF 1.5)* controls the playing of multimedia content (see Section 9.1, "Multimedia"). This action can be used in the following ways:

- To begin the playing of a rendition object (see Section 9.1.2, "Renditions"), associating it with a screen annotation (see "Screen Annotations" on page 639). The screen annotation specifies where the rendition is played unless otherwise specified.
- To stop, pause, or resume a playing rendition.
- To trigger the execution of a JavaScript script that may perform custom operations.

TABLE 8.64 Additional entries specifi

KEY	TYPE	VALUE
S	name	*(Required)* The type of action that thi rendition action.
R	dictionary	(*Required when **OP** is present with a val* ject (see Section 9.1.2, "Renditions").
AN	dictionary	(*Required if **OP** is present with a value* reference to a screen annotation (see "
OP	integer	(*Required if **JS** is not present; otherwis* action is triggered. Valid values are: 0 If no rendition is associated wi dition specified by **R**, associat ready associated with the ann associated with the annotation 1 Stop any rendition being play by **AN**, and remove the association. If no rendition is being played, there is no effect. 2 Pause any rendition being played in association with the annotation specified by **AN**. If no rendition is being played, there is no effect. 3 Resume any rendition being played in association with the annotation specified by **AN**. If no rendition is being played or the rendition is not paused, there is no effect. 4 Play the rendition specified by **R**, associating it with the annotation specified by **AN**. If a rendition is already associated with the annotation, resume the rendition if it is paused; otherwise, do nothing.
JS	text string or stream	(*Required if **OP** is not present; otherwise optional*) A text string or stream containing a JavaScript script to be executed when the action is triggered.

Either the **JS** entry or the **OP** entry must be present. If both are present, **OP** is considered a fallback to be executed if the viewer application is unable to execute

Before a rendition action is executed, the viewer
the **P** entry of the screen annotation dictionary r
that the annotation is present in the page object's

A rendition may play in the rectangle occupied b
annotation itself is not visible; for example, if its
ble 8.16) are set. If a screen annotation is not vi
page is not being displayed by the viewer, the re
may become visible if the view changes, such as

Transition Actions

A *transition action (PDF 1.5)* can be used to cont
actions. As discussed in Section 8.5.1, "Action D
action dictionary can specify a sequence of acti
normally suspend drawing when such a seque
when it ends. If a transition action is present duri
render the state of the page viewing area as it exist
action and display it using a transition specified in the action dictionary (see Table 8.65). Once this transition completes, drawing should be suspended again.

TABLE 8.65 Additional entries specific to a transition action

KEY	TYPE	VALUE
S	name	*(Required)* The type of action that this dictionary describes; must be **Trans** for a transition action.
Trans	dictionary	*(Required)* The transition to use for the update of the display (see Table 8.13).

Go-To-3D-View Actions

A *go-to-3D-view action (PDF 1.6)* identifies a 3D annotation and specifies a view for the annotation to use (see Section 9.5, "3D Artwork"). Table 8.66 shows the entries in a go-to-3D-view action dictionary.

K

S

KEY	TYPE	VALUE
TA	dictionary	*(Required)* The target annotation for
V	(various)	*(Required)* The view to use. It can be • A 3D view dictionary (see Section • An integer specifying an index i 9.35). • A text string matching the **IN** entr 9.39). • A name that indicates the first (**F** entries in the **VA** array; see discus

The **V** entry selects the view to apply to the anno may be one of the predefined views specified b (see Table 9.35) or a unique view specified here.

If the predefined view is specified by the names **N** (next) or **P** (previous), it should be interpreted in the following way:

- When the last view applied was specified by means of the **VA** array, **N** and **P** indicate the next and previous entries, respectively, in the **VA** array (wrapping around if necessary).
- When the last view was not specified by means of **VA**, using **N** or **P** should result in reverting to the default view.

8.6 Interactive Forms

An *interactive form (PDF 1.2)*—sometimes referred to as an *AcroForm*—is a collection of *fields* for gathering information interactively from the user. A PDF document may contain any number of fields appearing on any combination of pages, all of which make up a single, global interactive form spanning the entire document. Arbitrary subsets of these fields can be imported or exported from the

Each field in a document's interactive form is
Section 8.6.2, "Field Dictionaries"). For purpos
fields can be organized hierarchically and can
cestors in the field hierarchy. A field's children i
widget annotations (see "Widget Annotations"
pearance on the page. A field whose children a
terminal field.

As a convenience, when a field has only a single
contents of the field dictionary and the annotati
notation Dictionaries") may be merged into a sin
that pertain to both a field and an annotation. (
the contents of the two kinds of dictionaries do n
fines an appearance stream, the appearance mu
current value as a field.

Note: *Fields containing text whose contents are n*
construct their appearance streams dynamically i
in an appearance dictionary; see "Variable Text" on page 677.

8.6.1 Interactive Form Dictionary

The contents and properties of a document's interactive form are defined by an *interactive form dictionary* that is referenced from the **AcroForm** entry in the document catalog (see Section 3.6.1, "Document Catalog"). Table 8.67 shows the contents of this dictionary.

TABLE 8.67 Entries in the interactive form dictionary

KEY	TYPE	VALUE
Fields	array	*(Required)* An array of references to the document's *root fields* (those with no ancestors in the field hierarchy).
NeedAppearances	boolean	*(Optional)* A flag specifying whether to construct appearance streams and appearance dictionaries for all widget annotations in the docu-

S

KEY	TYPE	VALUE
CO	array	*(Required if any fields* *naries containing a* ***C*** *e* field dictionaries with *der* in which their val field changes (see Secti
DR	dictionary	*(Optional)* A resource tionaries") containing or spaces) to be use minimum, this diction source name and font (See implementation n
DA	string	*(Optional)* A documen able text fields (see "Va
Q	integer	*(Optional)* A documen able text fields (see "Va
XFA	stream or array	*(Optional; PDF 1.5)* A stream or array containing an *XFA resource*, whose format is described by the *Data Package (XDP) Specification*. (see the Bibliography). The value of this entry must be either a stream representing the entire contents of the XML Data Package or an array of text string and stream pairs representing the individual packets comprising the XML Data Package. See Section 8.6.7, "XFA Forms," for more information. ***Note:*** *In the original version of the PDF 1.5 specification, the value of this entry was defined as a stream only; see implementation note 115 in Appendix H.*

The value of the interactive form dictionary's **SigFlags** entry is an unsigned 32-bit integer containing flags specifying various document-level characteristics related to signature fields (see "Signature Fields" on page 695). Bit positions within the flag word are numbered from 1 (low-order) to 32 (high-order). Table 8.68 shows

TABLE 8.68 Signature

BIT POSITION	NAME	MEANING
1	SignaturesExist	If set, the document contai viewer application to enab pushbuttons) related to si entire document for the pr
2	AppendOnly	If set, the document contai is saved (written) in a way an incremental update. M mation to the end of the dating Example"). Viewer requesting a full save with will be invalidated and re with the operation.

8.6.2 Field Dictionaries

Each field in a document's interactive form is defined by a *field dictionary*, which must be an indirect object. The field dictionaries may be organized hierarchically into one or more tree structures. Many field attributes are *inheritable*, meaning that if they are not explicitly specified for a given field, their values are taken from those of its parent in the field hierarchy. Such inheritable attributes are designated as such in the tables below. The designation *(Required; inheritable)* means that an attribute must be defined for every field, whether explicitly in its own field dictionary or by inheritance from an ancestor in the hierarchy. Table 8.69 shows those entries that are common to all field dictionaries, regardless of type. Entries that pertain only to a particular type of field are described in the relevant sections below.

TABLE 8.69 Entries common to all

KEY	TYPE	VALUE
FT	name	*(Required for terminal fields; i* describes: **Btn** Button (see "Butt **Tx** Text (see "Text Fi **Ch** Choice (see "Cho **Sig** *(PDF 1.3)* Signat ***Note:*** *This entry may be prese are fields) to provide an inherit not logically have a type of its tributes that are intended for d*
Parent	dictionary	*(Required if this field is the chi wise)* The field that is the i whose **Kids** array includes this is, it can be included in the **Ki**
Kids	array	*(Sometimes required, as described below)* An array of indirect references to the immediate children of this field. In a non-terminal field, the **Kids** array is required to refer to field dictionaries that are immediate descendants of this field. In a terminal field, the **Kids** array ordinarily must refer to one or more separate widget annotations that are associated with this field. However, if there is only one associated widget annotation, and its contents have been merged into the field dictionary, **Kids** must be omitted.
T	text string	*(Optional)* The partial field name (see "Field Names," below; see also implementation notes 116 and 117 in Appendix H).
TU	text string	*(Optional; PDF 1.3)* An alternate field name to be used in place of the actual field name wherever the field must be identified in the user interface (such as in error or status messages referring to the field). This text is also useful when extracting the document's contents in support of accessibility to users with disabilities or for other purposes (see Section 10.8.2, "Alternate Descriptions").
T		
F		

KEY	TYPE	VALUE
V	(various)	*(Optional; inheritable)* The fi the field type. See the descrip mation.
DV	(various)	*(Optional; inheritable)* The d reset-form action is executed format of this value is the sam
AA	dictionary	*(Optional; PDF 1.2)* An add behavior in response to vari Events"). This entry has exa annotation dictionary (see Se

The value of the field dictionary's **Ff** entry is an ing flags specifying various characteristics of t flag word are numbered from 1 (low-order) to 3 in Table 8.70 are common to all types of fields. field types are discussed in the sections describi bits are reserved and must be set to 0.

TABLE 8.70 Field flags common to all field types

BIT POSITION	NAME	MEANING
1	ReadOnly	If set, the user may not change the value of the field. Any associated widget annotations will not interact with the user; that is, they will not respond to mouse clicks or change their appearance in response to mouse motions. This flag is useful for fields whose values are computed or imported from a database.
2	Required	If set, the field must have a value at the time it is exported by a submit-form action (see "Submit-Form Actions" on page 703).
3	NoExport	If set, the field must not be exported by a submit-form action (see "Submit-Form Actions" on page 703).

ancestors. For a field with no parent, the partial
same. For a field that is the child of another f
formed by appending the child field's partial na
name, separated by a period (.):

parent's_full_name.child's_partial_name

For example, if a field with the partial field nam
partial name is Address, which in turn has a chil
the fully qualified name of this last field is

PersonalData.Address.ZipCode

Thus, all fields descended from a common an
qualified field name as a common prefix in their

It is possible for different field dictionaries to h
name if they are descendants of a common ance
partial field names (**T** entries) of their own. Such field dictionaries are different representations of the same underlying field; they should differ only in properties that specify their visual appearance. In particular, field dictionaries with the same fully qualified field name must have the same field type (**FT**), value (**V**), and default value (**DV**).

Variable Text

When the contents and properties of a field are known in advance, its visual appearance can be specified by an appearance stream defined in the PDF file (see Section 8.4.4, "Appearance Streams," and "Widget Annotations" on page 640). In some cases, however, the field may contain text whose value is not known until viewing time. Examples include text fields to be filled in with text typed by the user from the keyboard and scrollable list boxes whose contents are determined interactively at the time the document is displayed.

TABLE 8.71 Additional entries common to all f

KEY	TYPE	VALUE
DA	string	*(Required; inheritable)* The *default ap* page-content graphics or text state ope text size and color.
Q	integer	*(Optional; inheritable)* A code specifyi used in displaying the text: 0 Left-justified 1 Centered 2 Right-justified Default value: 0 (left-justified).
DS	text string	*(Optional; PDF 1.5)* A default style stri 680.
RV	text string or text stream	*(Optional; PDF 1.5)* A rich text string, a

The new appearance stream becomes the normal appearance (**N**) in the appearance dictionary associated with the field's widget annotation (see Table 8.19 on page 614). (If the widget annotation has no appearance dictionary, the viewer application must create one and store it in the annotation dictionary's **AP** entry.)

In PDF 1.5, form fields that have the RichText flag set (see Table 8.77) specify formatting information as described in "Rich Text Strings" on page 680. For these fields, the conventions described below are not used, and the entire annotation appearance is regenerated each time the value is changed.

For non-rich text fields, the appearance stream—which, like all appearance streams, is a form XObject—has the contents of its form dictionary initialized as follows:

- The resource dictionary (**Resources**) is created using resources from the inter-

the dimensions of the annotation rectangle (th
tation dictionary).

- All other entries in the appearance stream's
default values (see Section 4.9, "Form XObject

The appearance stream includes the following s
represents the portion of the stream that draws t

Example 8.14

```
/Tx BMC
   q
      …Any required graphics state changes, such as
      BT
         …Default appearance string (DA)…
         …Text-positioning and text-showing operat
      ET
   Q
EMC                                          % End marked content
```

The **BMC** (begin marked content) and **EMC** (end marked content) operators are discussed in Section 10.5, "Marked Content". **q** (save graphics state) and **Q** (restore graphics state) are discussed in Section 4.3.3, "Graphics State Operators". **BT** (begin text object) and **ET** (end text object) are discussed in Section 5.3, "Text Objects." See Example 8.18 for an example.

The default appearance string (**DA**) contains any graphics state or text state operators needed to establish the graphics state parameters, such as text size and color, for displaying the field's variable text. Only operators that are allowed within text objects may occur in this string (see Figure 4.1 on page 197). At a minimum, the string must include a **Tf** (text font) operator along with its two operands, *font* and *size*. The specified *font* value must match a resource name in the **Font** entry of the default resource dictionary (referenced from the **DR** entry of the interactive form dictionary; see Table 8.67). A zero value for *size* means that the font is to be *auto-sized*: its size is computed as a function of the height of the annotation rectangle.

viewer should insert one in the appearance stre
and vertical translation components) after the d
fore the text-positioning and text-showing opera

To update an existing appearance stream to refl
application should first copy any needed resour
tionary (see Table 8.67) into the stream's **Reso
Resources dictionaries contain resources with th
the **Resources** dictionary should be left intact, *n*
ing value from the **DR** dictionary.) The viewer ap
existing contents of the appearance stream fro
with the corresponding new contents as shown
appearance stream contains no marked conten
should be appended to the end of the original st
note 119 in Appendix H.

Rich Text Strings

Beginning with PDF 1.5, the text contents of variable text form fields, as well as markup annotations, can include formatting (style) information. These *rich text strings* are fully-formed XML documents that conform to the rich text conventions specified for the XML Forms Architecture (XFA) specification, which is itself a subset of the XHTML 1.0 specification, augmented with a restricted set of CSS2 style attributes (see the Bibliography for references to all these standards).

Table 8.72 lists the XHTML elements that can appear in rich text strings. The <body> element is the root element; its required attributes are listed in Table 8.73. Other elements (<p> and) contain enclosed text that may take style attributes, which are listed in Table 8.74. These style attributes are CSS inline style property declarations of the form *name*:*value*, with each declaration separated by a semicolon, as illustrated in Example 8.15 on page 684.

In PDF 1.6, PDF supports the rich text elements and attributes specified in the *XML Forms Architecture (XFA) Specification, 2.2* (see Bibliography). These rich

TABLE 8.72 XHTML elements used

ELEMENT	DESCRIPTION
<body>	The element at the root of the XML document element.
<p>	Encloses text that is interpreted as a paragraph 8.74.
<i>	Encloses text that is displayed in an italic font.
<b>	Encloses text that is displayed in a bold font.
	Groups text solely for the purpose of applying s

TABLE 8.73 Attributes of the <

ATTRIBUTE	DESCRIPTION
xmlns	The default namespaces for elements within the rich text string. Must be xmlns="http://www.w3.org/1999/xhtml" xmlns:xfa="http://www.xfa.org/schema/xfa-data/1.0".
xfa:contentType	Must be "text/html".
xfa:APIVersion	A string that identifies the software used to generate the rich text string. It must be of the form software_name:software_version, where • software_name identifies the software by name. It must not contain spaces. • software_version identifies the version of the software. It consists of a series of integers separated by decimal points. Each integer is a version number, the leftmost value being a major version number, with values to the right increasingly minor. When comparing strings, the versions are compared in order. For example "5.2" is less than "5.13" because 2 is less than 13; the string is not treated as a decimal number. When comparing strings with different numbers of sections, the string with fewer sections is implicitly padded on the right with sections containing "0" to make the number of sections equivalent.
xfa:spec	The version of the *XML Forms Architecture (XFA)* specification to which the rich text

TABLE 8.74 CSS2 style attributes use

ATTRIBUTE	VALUE	DESCRIPTION
text-align	keyword	Horizontal alignment. Possib
vertical-align	decimal	An amount by which to adju value indicates a superscript; is of the form <*decimal num* lowed by "pt". Examples: -3pt,
font-size	decimal	The font size of the enclosed <*decimal number*>pt.
font-style	keyword	Specifies whether the enclos italic (oblique) font. Possible
font-weight	keyword	The weight of the font for t 100, 200, 300, 400, 500, 600, ***Note:*** *normal is equivalent to*
font-family	list	A font name or list of font names to be used to display the enclosed text. (If a list is provided, the first one containing glyphs for the specified text is used.)
font	list	A shorthand CSS font property of the form font:<*font-style*> <*font-weight*> <*font-size*> <*font-family*>
color	RGB value	The color of the enclosed text. It can be in one of two forms: • #*rrggbb* with a 2-digit hexadecimal value for each component • rgb*(rrr,ggg,bbb)* with a decimal value for each component. ***Note:*** *Although the values specified by the color property are interpreted as sRGB values, they are transformed into values in a non-ICC based color space when used to generate the annotation's appearance.*
text-decoration	keyword	One of the following keywords: • underline: The enclosed text should be underlined. • line-through: The enclosed text should have a line drawn through it.

f

Rich text strings are specified by the **RV** entry of
ies (see Table 8.71) and the **RC** entry of markup a
8.21). Rich text strings may be packaged as *text*
Streams"). Form fields using rich text streams s
set (see Table 8.77).

A *default style string* is specified by the **DS** entry
ble 8.25) or variable text form fields (see Table 8
fault values for style attributes, which are used fo
explicitly specified for the annotation or field.
are legal in the default style string. This string, in
used to generate the appearance. The following
compliant viewers: the **Contents** entry for annot
annotations, and the **V**, **DA**, and **Q** entries for for

Note: *Markup annotations other than free text a*
tions" on page 616) do not use a default style stri
implemented using platform controls requiring the
propriate system font for display.

When a form field or annotation contains rich text strings, the *flat text* (character data) of the string should also be preserved (in the **V** entry for form fields and the **Contents** entry for annotations). This enables older viewer applications to read and edit the data (although with loss of formatting information). The **DA** entry should be written out as well when the file is saved.

If a document containing rich text strings is edited in a viewer that does not support PDF 1.5, the rich text strings remain unchanged (because they are unknown to the viewer), even though the corresponding flat text may have changed. When a viewer that supports PDF 1.5 reads a rich text string from a document, it must check whether the corresponding flat text has changed by using the following procedure:

1. Create a new flat text string containing the character data from the rich text string. Character references (such as
) should be converted to their char-

2. If either of the values uses UTF-16 encoding, 16 if necessary.
3. Compare the resulting strings.

If the strings are unequal, it is assumed the fiel viewer, and a new rich text string should be crea

When a rich text string specifies font attributes, t font name selection as described in section 15.3 Bibliography). It is strongly recommended that in the default resources dictionary, as specified Implementation note 120 in Appendix H.

The following example illustrates the entries in for rich text. The **DS** entry specifies the default paragraphs of rich text: the first paragraph speci fault font; the second paragraph changes the font

Example 8.15

```
/DS (font: 18pt Arial)                % Default style string using an abbreviated font
                                      % descriptor to specify 18pt text using an Arial font

/RV (<?xml version="1.0"?><body xmlns="http://www.w3.org/1999/xtml"
      xmlns:xfa="http://www.xfa.org/schema/xfa-data/1.0/"
      xfa:contentType="text/html" xfa:APIVersion="Acrobat:8.0.0" xfa:spec="2.4">
      <p style="text-align:left">
         <b>
            <i>
               Here is some bold italic text
            </i>
         </b>
      </p>
      <p style= "font-size:16pt">
         This text uses default text state parameters but changes the font size to 16.
      </p>
   </body> )
```

8.6.3 Field Types

Interactive forms support the following field typ

- *Button fields* represent interactive controls o manipulate with the mouse. They include *pus buttons*.
- *Text fields* are boxes or spaces in which the u board.
- *Choice fields* contain several text items, at most the field value. They include scrollable *list box*
- *Signature fields* represent electronic signatures a user and the validity of the document's conte

The following sections describe each of these fi may be added in the future.

Button Fields

A *button field* (field type **Btn**) represents an interactive control on the screen that the user can manipulate with the mouse. There are three types of button fields:

- A *pushbutton* is a purely interactive control that responds immediately to user input without retaining a permanent value (see "Pushbuttons" on page 686).
- A *check box* toggles between two states, on and off (see "Check Boxes" on page 686).
- *Radio button fields* contain a set of related buttons that can each be on or off. Typically, at most one radio button in a set may be on at any given time, and selecting any one of the buttons automatically deselects all the others. (There are exceptions to this rule, as noted in "Radio Buttons" on page 688.)

The various types of button fields are distinguished by flags in the **Ff** entry, as

TABLE 8.75 Field flags specific t

BIT POSITION	NAME	MEANING
15	NoToggleToOff	*(Radio buttons only)* If set, times; clicking the currentl the selected button deselec
16	Radio	If set, the field is a set of This flag is meaningful onl
17	Pushbutton	If set, the field is a pushbut
26	RadiosInUnison	*(PDF 1.5)* If set, a group o use the same value for the one is checked, they are al clusive (the same behavior

Pushbuttons

The simplest type of field is a *pushbutton field*, which has a field type of **Btn** and the Pushbutton flag (see Table 8.75) set. Because this type of button retains no permanent value, it does not use the **V** and **DV** entries in the field dictionary (see Table 8.69 on page 675).

Check Boxes

A *check box field* represents one or more check boxes that toggle between two states, on and off, when manipulated by the user with the mouse or keyboard. Its field type is **Btn** and its Pushbutton and Radio flags (see Table 8.75) are both clear. Each state can have a separate appearance, which is defined by an appearance stream in the appearance dictionary of the field's widget annotation (see Section 8.4.4, "Appearance Streams"). The appearance for the off state is optional but, if present, must be stored in the appearance dictionary under the name Off. The recommended (but not required) name for the on state is Yes.

Example 8.16

```
1 0 obj
   << /FT /Btn
      /T (Urgent)
      /V /Yes
      /AS /Yes
      /AP << /N << /Yes 2 0 R /Off 3 0 R>>
   >>
endobj

2 0 obj
   << /Resources 20 0 R
      /Length 104
   >>

stream
   q
      0 0 1 rg
      BT
         /ZaDb 12 Tf
         0 0 Td
         (8) Tj
      ET
   Q
endstream
endobj

3 0 obj
   << /Resources 20 0 R
      /Length 104
   >>

stream
   q
      0 0 1 rg
      BT
         /ZaDb 12 Tf
         0 0 Td
         (8) Tj
```

Beginning with PDF 1.4, the field dictionary fo
contains an optional **Opt** entry (see Table 8.76
strings representing the export value of each an
the following purposes:

- To represent the export values of check box an
in writing systems. Because name objects in th
ited to **PDFDocEncoding**, they cannot represe

- To allow radio buttons or check boxes to be
they have the same export value.

  An example is a group of check boxes that a
page, and the desired behavior is that when
sponding boxes on each of the other pages is a
the corresponding check boxes is a widget in t

  Note: *For radio buttons, the same behavior occ
is set. If it is not set, at most one radio button i
implementation note 121 in Appendix H.*

TABLE 8.76 Additional entry specific to check box and radio button fields

KEY	TYPE	VALUE
Opt	array of text strings	*(Optional; inheritable; PDF 1.4)* An array containing one entry for each widget annotation in the **Kids** array of the radio button or check box field. Each entry is a text string representing the on state of the corresponding widget annotation. When this entry is present, the names used to represent the on state in the **AP** dictionary of each annotation are computer-generated numbers equivalent to the numerical position (starting with **0**) of the annotation in the **Kids** array. This allows distinguishing between the annotations even if two or more of them have the same value in the **Opt** array. For example, two radio buttons may have the same on state, but if the RadiosInUnison flag is not set, only one of them at a time can be checked by the user.

Radio Buttons

most one button in the on state at any given time
tomatically deselects all the others.

Note: *An exception occurs when multiple radio b*
state and the RadiosInUnison flag is set. In that c
turns on all of them.

The field type is **Btn**, the Pushbutton flag (see Ta
the Radio flag is set. This type of button field h
ToOff, which specifies, if set, that exactly one of
ed at all times. In this case, clicking the currentl
the NoToggleToOff flag is clear, clicking the sel
no button selected.

The **Kids** entry in the radio button field's field di
675) holds an array of widget annotations repre
the set. The parent field's **V** entry holds a name
pearance state of whichever child field is current
ue for this entry is Off. Example 8.17 shows the object definitions for a set of radio buttons.

Example 8.17

```
10  0  obj                                  % Radio button field
   <<  /FT  /Btn
       /Ff  …                               % … Radio flag = 1, Pushbutton = 0 …
       /T  (Credit card)
       /V  /MasterCard
       /Kids  [  11 0 R
                 12 0 R
              ]
   >>
endobj

11  0  obj                                  % First radio button
   <<  /Parent  10 0 R
       /AS  /MasterCard
```

```
12 0 obj                                        %
   << /Parent 10 0 R
      /AS /Off
      /AP << /N << /Visa 8 0 R
                   /Off 9 0 R
                >>
         >>
   >>
endobj

8 0 obj                                         %
   << /Resources 20 0 R
      /Length 104
   >>
stream
   q
      0 0 1 rg
      BT
         /ZaDb 12 Tf
         0 0 Td
         (8) Tj
      ET
   Q
endstream
endobj

9 0 obj                                         % Appearance stream for "off" state
   << /Resources 20 0 R
      /Length 104
   >>
stream
   q
      0 0 1 rg
      BT
         /ZaDb 12 Tf
         0 0 Td
         (4) Tj
      ET
```

using Unicode encoding for non-Latin characte
array of text strings corresponding to the widget
dividual buttons in the field's **Kids** array.

Text Fields

A *text field* (field type **Tx**) is a box or space in w
the keyboard. The text may be restricted to a si
span multiple lines, depending on the setting of t
tionary's **Ff** entry. Table 8.77 shows the flags pert

TABLE 8.77 Field flags specifi

BIT POSITION	NAME	MEANING
13	Multiline	If set, the field can contain restricted to a single line.
14	Password	If set, the field is intended be echoed visibly to the screen. Characters typed from the keyboard should instead be echoed in some unreadable form, such as asterisks or bullet characters. To protect password confidentiality, viewer applications should never store the value of the text field in the PDF file if this flag is set.
21	FileSelect	*(PDF 1.4)* If set, the text entered in the field represents the pathname of a file whose contents are to be submitted as the value of the field.
23	DoNotSpellCheck	*(PDF 1.4)* If set, text entered in the field is not spell-checked.
24	DoNotScroll	*(PDF 1.4)* If set, the field does not scroll (horizontally for single-line fields, vertically for multiple-line fields) to accommodate more text than fits within its annotation rectangle. Once the field is full, no further text is accepted.
25	Comb	*(PDF 1.5)* Meaningful only if the **MaxLen** entry is present in the text field dictionary (see Table 8.78) and if the Multiline, Password, and FileSelect flags are clear. If set, the field is automatically divided into as many equally

2

The field's text is held in a text string (or, beginni
V (value) entry of the field dictionary. The cont
are used to construct an appearance stream for
under "Variable Text" on page 677. The text is
size, color, and so forth), as specified by the **DA** (

If the FileSelect flag *(PDF 1.4)* is set, the field fu
this case, the field's text represents the pathname
submitted as the field's value:

- For fields submitted in HTML Form format, content type multipart/form-data, as describ *purpose Internet Mail Extensions (MIME), Part Bodies* (see the Bibliography).
- For Forms Data Format (FDF) submission, th field dictionary (see "FDF Fields" on page 71 3.10, "File Specifications") identifying the sele
- XML format is not supported for file-select controls; therefore, no value is submitted in this case.

Besides the usual entries common to all fields (see Table 8.69 on page 675) and to fields containing variable text (see Table 8.71), the field dictionary for a text field can contain the additional entry shown in Table 8.78.

TABLE 8.78 Additional entry specific to a text field

KEY	TYPE	VALUE
MaxLen	integer	*(Optional; inheritable)* The maximum length of the field's text, in characters.

Example 8.18 shows the object definitions for a typical text field.

Example 8.18

```
6 0 obj
   << /FT /Tx
      /Ff …                                    % Set Multiline flag
```

```
5 0 obj
   << /Resources 21 0 R
      /Length 172
   >>
stream
   /Tx BMC
      q
         BT
            0 0 1 rg
            /Ti 12 Tf
            1 0 0 1 100 100 Tm
            0 0 Td
            (The quick brown fox ) Tj
            0 –13 Td
            (ate the lazy mouse.) Tj
         ET
      Q
   EMC
endstream
endobj
```

Choice Fields

A *choice field* (field type **Ch**) contains several text items, one or more of which may be selected as the field value. The items may be presented to the user in either of two forms:

- A scrollable *list box*
- A *combo box* consisting of a drop-down list optionally accompanied by an editable text box in which the user can type a value other than the predefined choices

TABLE 8.79 Field flags specific to choice fields

BIT POSITION	NAME	MEANING
1		
1		

BIT POSITION	NAME	MEANING
20	Sort	If set, the field's option is intended for use by cations. Viewers shou which they occur in th
22	MultiSelect	*(PDF 1.4)* If set, more lected simultaneously; be selected.
23	DoNotSpellCheck	*(PDF 1.4)* If set, text e is meaningful only if t
27	CommitOnSelChange	*(PDF 1.5)* If set, the n made with the pointi perform an action onc to exit the field. If clea exits the field.

The various types of choice fields are distinguished by flags in the **Ff** entry, as shown in Table 8.79. Table 8.80 shows the field dictionary entries specific to choice fields.

TABLE 8.80 Additional entries specific to a choice field

KEY	TYPE	VALUE
Opt	array	*(Optional)* An array of options to be presented to the user. Each element of the array is either a text string representing one of the available options or an array consisting of two text strings: the option's export value and the text to be displayed as the name of the option (see implementation note 122 in Appendix H). If this entry is not present, no choices should be presented to the user.
TI	integer	*(Optional)* For scrollable list boxes, the *top index* (the index in the **Opt** array of the first option visible in the list). Default value: 0.
I	array	*(Sometimes required, otherwise optional; PDF 1.4)* For choice fields that allow multiple selection (MultiSelect flag set), an array of integers, sorted in ascending order, represent-

The **Opt** array specifies the list of options in the
resented by a text string to be displayed on the
array contains either this text string by itself or
ond element is the text string and whose first ele
the export value to be used when exporting int
document.

The field dictionary's **V** (value) entry (see Table
item or items currently selected in the choice fiel
tiple selection—that is, if the MultiSelect flag *(P*
selection is supported but only one item is cur
representing the name of the selected item, as g
array. If multiple items are selected, **V** is an array
sented in the **Opt** array by a two-element array,
the two array elements.) The default value of **V**
currently selected.

Example 8.19 shows a typical choice field definit

Example 8.19

```
<< /FT /Ch
   /Ff …
   /T (Body Color)
   /V (Blue)
   /Opt [ (Red)
          (My favorite color)
          (Blue)
        ]
>>
```

Signature Fields

A *signature field (PDF 1.3)* is a form field that contains a digital signature (see Section 8.7, "Digital Signatures"). The field dictionary representing a signature

Filling in (signing) the signature field entails u
usually also the **AP** entry of the associated widge
ture field typically exports the **T**, **V**, and **AP** entrie

Like any other field, a signature field may actuall
tation dictionary containing entries pertaining t
(see "Widget Annotations" on page 640). The an
a dictionary gives the position of the field on its
intended to be visible should have an annotatio
and width.

The appearance dictionary (**AP**) of a signature
the field's visual appearance on the page (
Streams"). Information about how Acrobat hand
is in the technical note *Digital Signature Appeara*

TABLE 8.81 Additional entries specifi

KEY	TYPE	VALUE
Lock	dictionary	*(Optional; must be an indirect reference; PDF 1.5)* A *signature field lock dictionary* that specifies a set of form fields to be locked when this signature field is signed. Table 8.82 lists the entries in this dictionary.
SV	dictionary	*(Optional; must be an indirect reference; PDF 1.5)* A *seed value dictionary* (see Table 8.83) containing information that constrains the properties of a signature that is applied to this field.

The value of the **SV** entry in the field dictionary is a seed value dictionary whose entries (see Table 8.83) provide constraining information that is to be used at the time the signature is applied. Its **Ff** entry specifies whether the other entries in the dictionary are required to be honored or whether they are merely recommendations.

Note: *The seed value dictionary may include seed values for private entries belonging to multiple handlers. A given handler should use only those entries that are per-*

TABLE 8.82 Entries in a signature fi

KEY	TYPE	VALUE
Type	name	*(Optional)* The type of PDF object th **SigFieldLock** for a signature field lock
Action	name	*(Required)* A name which, in conjunc should be locked. Valid values are: All — All fields in the docu Include — All fields specified in Exclude — All fields except those
Fields	array	*(Required if the value of **Action** is Includ* field names.

TABLE 8.83 Entries in a signature field

KEY	TYPE	VALUE
Type	name	*(Optional)* The type of PDF object that this dictionary describes; if present, must be **SV** for a seed value dictionary.
Filter	name	*(Optional)* The signature handler to be used to sign the signature field. Beginning with PDF 1.7, if **Filter** is specified and the **Ff** entry indicates this entry is a required constraint, then the signature handler specified by this entry must be used when signing; otherwise, signing must not take place. If **Ff** indicates that this is an optional constraint, this handler should be used if it is available. If it is not available, a different handler can be used instead.
SubFilter	array	*(Optional)* An array of names indicating encodings to use when signing. The first name in the array that matches an encoding supported by the signature handler should be the encoding that is actually used for signing. If **SubFilter** is specified and the **Ff** entry indicates that this entry is a required constraint, then the first matching encodings must be used when signing; otherwise, signing must not take place. If **Ff** indicates that this is an optional constraint, then the

KEY	TYPE	VALUE
DigestMethod	array	(*Optional; PDF 1.7*) An array o to use while signing. The valid **RIPEMD160**. The default value i ***Note:*** *This property is only ap RSA public/private keys. If it c rithm is always **SHA1** and this a*
V	real	*(Optional)* The minimum requ dictionary parser. A value of 1 nize all seed value dictionary e that it must be able to recogni PDF 1.7 and earlier. The **Ff** entry indicates whether ***Note:*** *The* PDF Reference fifth *cates that the* **V** *entry is of type i*
Cert	dictionary	*(Optional)* A *certificate seed value dictionary* (see Table 8.84) containing information about the certificate to be used when signing.
Reasons	array	*(Optional)* An array of text strings that specifying possible reasons for signing a document. If specified, the reasons supplied in this entry replace those used by viewer applications. The **Ff** entry specifies whether one of the reasons in the array must be used in the signature. • If the **Reasons** array is provided and the **Ff** entry indicates that **Reasons** is a required constraint, one of the reasons in the array must be used for the signature dictionary; otherwise, signing must not take place. If the **Ff** entry indicates **Reasons** is an optional constraint, one of the reasons in the array can be chosen or a custom reason can be provided. • If the **Reasons** array is omitted or contains a single 0-character length string and the **Ff** entry indicates that **Reasons** is a required constraint, the **Reason** entry must be omitted from the signature dictionary (see Table 8.102).
MDP	dictionary	*(Optional; PDF 1.6)* A dictionary containing a single entry whose key is **P** and whose value is an integer between 0 and 3. A value of 0 defines the signature as

KEY	TYPE	VALUE
TimeStamp	dictionary	*(Optional; PDF 1.6)* A time sta **URL** An ASCII string s providing a time s *X.509 Public Key I* liography). **Ff** An integer whose time stamp) or 0 stamp). Default va
LegalAttestation	array	*(Optional; PDF 1.6)* An array tions (see Section 8.7.4, "Legal sponding flag in the **Ff** entry in
AddRevInfo	boolean	(*Optional; PDF 1.7*) A flag ind carried out. If **AddRevInfo** is tr ing additional tasks when signi • Perform revocation checking of the certificate (and the corresponding issuing certificates) used to sign. • Include the revocation information within the signature value. A value of true is relevant only if **SubFilter** is **adbe.pkcs7.detached** or **adbe.pkcs7.sha1**. If **SubFilter** is **x509.rsa_sha1**, this entry must be omitted or set to false; otherwise, the signature process may fail. If **AddRevInfo** is true and the **Ff** entry indicates this is a required constraint, then the tasks described above must be performed. If they cannot be performed, then signing must fail. Default value: false
Ff	integer	*(Optional)* A set of bit flags specifying the interpretation of specific entries in this dictionary. A value of 1 for the flag indicates that the associated entry is a required constraint. A value of 0 indicates that the associated entry is an optional constraint. Bit positions are 1 (**Filter**); 2 (**SubFilter**); 3 (**V**); 4 (**Reasons**); 5 (**LegalAttestation**); 6(**AddRevInfo**); and 7(**DigestMethod**). Default value: 0.

TABLE 8.84 Entries in a certificate se

KEY	TYPE	VALUE
Type	name	*(Optional)* The type of PDF o must be **SVCert** for a certificate
Subject	array	*(Optional)* An array of byte str cates that are acceptable for sig 3280, *Internet X.509 Public Key cation List (CRL) Profile* (see th flag in the **Ff** entry indicates w
SubjectDN	array of dictionaries	(*Optional; PDF 1.7*) An array key value pairs, that specify th be present within the certificat cate must at a minimum conta That is, the certificate can co guished Name is described in any legal attribute identifier. A 'email', '2.5.4.43' and always contain characters in the set a-z, A-Z, 0-9 and '.'. Values are text strings. An example dictionary is <</cn (John Smith) /1.5.4.43 (JS)>>. The value of the corresponding flag in the **Ff** entry indicates whether this entry is a required constraint.

KEY	TYPE	VALUE
KeyUsage	array of ASCII strings	(*Optional; PDF 1.7*) An array acceptable key-usage extensio Multiple strings specify a rang usage extension is described in Each character in a string repr characters indicates the key-u ninth characters in the array, fr the following key-usage extens 1 digitalSignature 2 non-Repudiation 3 keyEncipherment Any additional characters are that are not one of the follow character values are supported: 0 Corresponding key-usage must not be set. 1 Corresponding key-usage must be set. X State of the corresponding key-usage does not matter. For example, the string values '1' and '1XXXXXXXX' represent settings where the key-usage type digitalSignature must be set and the state of all other key-usage types do not matter. The value of the corresponding flag in the **Ff** entry indicates whether this is a required constraint.
Issuer	array	*(Optional)* An array of byte strings containing DER-encoded X.509v3 certificates of acceptable issuers. If the signer's certificate chains up to any of the specified issuers (either directly or indirectly), the certificate is considered acceptable for signing. The value of the corresponding flag in the **Ff** entry indicates whether this is a required constraint.
OID	array	*(Optional)* An array of byte strings that contain Object Identifiers (OIDs) of the certificate policies that must be present in the signing certificate. An exam-

KEY	TYPE	VALUE
URL	ASCII string	*(Optional)* A URL, the use for
URLType	Name	(*Optional; PDF 1.7*) A name i standard uses and there can be following value specifies a vali **Browser** – The URL refere browser to allow enroll is not found. The **Ff** att The following value specifies for use by Adobe Systems: **ASSP** – The URL reference server-based signing. I required constraint, thi must come from this s Third parties can extend the u ues, which must conform to the guidelines described in Appendix E. The default value is **Browser**.
Ff	integer	*(Optional)* A set of bit flags specifying the interpretation of specific entries in this dictionary. A value of 1 for the flag means that a signer is required to use only the specified values for the entry. A value of 0 means that other values are permissible. Bit positions are 1 (**Subject**); 2 (**Issuer**); 3 (**OID**); 4 (**SubjectDN**); 5 (**Reserved**); 6 (**KeyUsage**); 7 (**URL**). Default value: 0.

8.6.4 Form Actions

Interactive forms support four special types of actions in addition to those described in Section 8.5.3, "Action Types":

- *Submit-form actions* transmit the names and values of selected interactive form

- *JavaScript actions (PDF 1.3)* cause a script to JavaScript interpreter.

Submit-Form Actions

A *submit-form action* transmits the names and v fields to a specified uniform resource locator (U Web server that will process them and send back action dictionary entries specific to this type of a

The value of the action dictionary's **Flags** entry i taining flags specifying various characteristics o the flag word are numbered from 1 (low-order shows the meanings of the flags; all undefined fl set to 0.

TABLE 8.85 Additional entries specific t

KEY	TYPE	VALUE
S	name	*(Required)* The type of action that this dictionary describes; must be **SubmitForm** for a submit-form action.
F	file specification	*(Required)* A URL file specification (see Section 3.10.4, "URL Specifications") giving the uniform resource locator (URL) of the script at the Web server that will process the submission.
Fields	array	*(Optional)* An array identifying which fields to include in the submission or which to exclude, depending on the setting of the Include/Exclude flag in the **Flags** entry (see Table 8.86). Each element of the array is either an indirect reference to a field dictionary or *(PDF 1.3)* a text string representing the fully qualified name of a field. Elements of both kinds may be mixed in the same array. If this entry is omitted, the Include/Exclude flag is ignored, and all fields in the document's interactive form are submitted except those whose NoExport flag (see Table 8.70 on page 676) is set. (Fields

F

—

TABLE 8.86 Flags for submit-

BIT POSITION	NAME	MEANING
1	Include/Exclude	If clear, the **Fields** a include in the submi the field hierarchy a If set, the **Fields** arr document's interacti **Fields** array and tho 676) is set.
2	IncludeNoValueFields	If set, all fields des Exclude flag are sub (**V** entry in the field field name is transm If clear, fields witho
3	ExportFormat	Meaningful only if t field names and values are submitted in HTML Form format. If clear, they are submitted in Forms Data Format (FDF); see Section 8.6.6, "Forms Data Format."
4	GetMethod	If set, field names and values are submitted using an HTTP GET request. If clear, they are submitted using a POST request. This flag is meaningful only when the ExportFormat flag is set; if ExportFormat is clear, this flag must also be clear.
5	SubmitCoordinates	If set, the coordinates of the mouse click that caused the submit-form action are transmitted as part of the form data. The coordinate values are relative to the upper-left corner of the field's widget annotation rectangle. They are represented in the data in the format *name*.x=*xval*&*name*.y=*yval* where *name* is the field's mapping name (**TM** in the field dictionary) if present; otherwise, *name* is the field name. If the value of the **TM** entry is a single space character, both the name and the dot following it are suppressed, resulting in the format

BIT POSITION	NAME	MEANING
6	XFDF	*(PDF 1.4)* Meaningf field names and valu
7	IncludeAppendSaves	*(PDF 1.4)* Meaningf Forms Data Format mat flags are clear). tents of all increme as contained in the Table 8.93 on page 7 cluded.
8	IncludeAnnotations	*(PDF 1.4)* Meaningf Forms Data Format mat flags are clear). up annotations in Annotations" on pa cluded.
9	SubmitPDF	*(PDF 1.4)* If set, the document is submitted as PDF, using the MIME content type application/pdf (described in Internet RFC 2045, *Multipurpose Internet Mail Extensions (MIME), Part One: Format of Internet Message Bodies*; see the Bibliography). If set, all other flags are ignored except GetMethod.
10	CanonicalFormat	*(PDF 1.4)* If set, any submitted field values representing dates are converted to the standard format described in Section 3.8.3, "Dates." (The interpretation of a form field as a date is not specified explicitly in the field itself but only in the JavaScript code that processes it.)
11	ExclNonUserAnnots	*(PDF 1.4)* Meaningful only when the form is being submitted in Forms Data Format (that is, when both the XFDF and ExportFormat flags are clear) and the IncludeAnnotations flag is set. If set, it includes only those markup annotations whose **T** entry (see Table 8.21) matches the name of the current user, as determined by the remote server to which the form is being submitted. (The **T** entry for markup annotations specifies the text label to be

BIT POSITION	NAME	MEANING
12	ExclFKey	*(PDF 1.4)* Meaningf Forms Data Format mat flags are clear).
14	EmbedForm	*(PDF 1.5)* Meaningf Forms Data Format mat flags are clear). specification contai PDF file from which

The set of fields whose names and values are t **Fields** array in the action dictionary (Table 8 Exclude and IncludeNoValueFields flags in the **F** ment of the **Fields** array identifies an interactive reference to its field dictionary or *(PDF 1.3)* by i "Field Names" on page 676). If the Include/Excl consists of all fields listed in the **Fields** array, alo fields in the field hierarchy. If the Include/Exclude flag is set, the submission consists of all fields in the document's interactive form *except* those listed in the **Fields** array.

Note: *The NoExport flag in the field dictionary's* ***Ff*** *entry (see Table 8.69 on page 675 and Table 8.70 on page 676) takes precedence over the action's* ***Fields*** *array and Include/Exclude flag. Fields whose NoExport flag is set are* never *included in a submit-form action.*

Field names and values may be submitted in any of the following formats, depending on the settings of the action's ExportFormat, SubmitPDF, and XFDF flags (see the Bibliography for references):

- HTML Form format (described in the *HTML 4.01 Specification*)
- Forms Data Format (FDF), which is described in Section 8.6.6, "Forms Data Format"; see also implementation note 123 in Appendix H.
-

The name submitted for each field is its fully qu
on page 676), and the value is specified by the **V**

Note: *For pushbutton fields submitted in FDF, the entry in the field's widget annotation dictionary. nary contains no* ***Fields*** *entry, such pushbutton fiel*

Fields with no value (that is, whose field dictio
are ordinarily not included in the submission. T
NoValueFields flag can override this behavior.
fields are included in the submission by name o

Reset-Form Actions

A *reset-form action* resets selected interactive fo
that is, it sets the value of the **V** entry in the field
(see Table 8.69 on page 675). If no default value i
removed. For fields that can have no value (suc
no effect. Table 8.87 shows the action dictionary entries specific to this type of action.

The value of the action dictionary's **Flags** entry is an unsigned 32-bit integer containing flags specifying various characteristics of the action. Bit positions within the flag word are numbered from 1 (low-order) to 32 (high-order). At the time of publication, only one flag is defined for this type of action; Table 8.88 shows its meaning. All undefined flag bits are reserved and must be set to 0.

TABLE 8.87 Additional entries specific to a reset-form action

KEY	TYPE	VALUE
S	name	*(Required)* The type of action that this dictionary describes; must be **ResetForm** for a reset-form action.