CHAPTER 8

Interactive Features

This chapter describes the PDF features that allow a user to interact with a document on the screen, using the mouse and keyboard (with the exception of multimedia features, which are described in Chapter 9, “Multimedia Features”):

- *Preference settings* to control the way the document is presented on the screen (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

The **ViewerPreferences** entry in a document’s catalog (see Section 3.6.1, “Document Catalog”) designates a *viewer preferences dictionary (PDF 1.2)* controlling the way the document is to be presented on the screen or in print. If no such dictionary is specified, viewing and printing applications should behave in accordance with their own current user preference settings. Table 8.1 shows the contents of the viewer preferences dictionary.

TABLE 8.1 Entries in a viewer preferences dictionary

KEY	TYPE	VALUE
HideToolbar	boolean	*(Optional)* A flag specifying whether to hide the viewer application's tool bars when the document is active. Default value: **false**.
HideMenubar	boolean	*(Optional)* A flag specifying whether to hide the viewer application's menu bar when the document is active. Default value: **false**.
HideWindowUI	boolean	*(Optional)* A flag specifying whether to hide user interface elements in the document's window (such as scroll bars and navigation controls), leaving only the document's contents displayed. Default value: **false**.
FitWindow	boolean	*(Optional)* A flag specifying whether to resize the document's window to fit the size of the first displayed page. Default value: **false**.
CenterWindow	boolean	*(Optional)* A flag specifying whether to position the document's window in the center of the screen. Default value: **false**.
DisplayDocTitle	boolean	*(Optional; PDF 1.4)* A flag specifying whether the window's title bar 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 R2L Right to left (including vertical writing systems, such as Chinese, Japanese, and Korean) This entry has no direct effect on the document's contents or page numbering but can be used to determine the relative positioning of pages when displayed side by side or printed *n*-up. Default value: L2R.

KEY	TYPE	VALUE
ViewArea	name	*(Optional; PDF 1.4)* The name of the page boundary representing the area of a page to be displayed when viewing the document on the screen. 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.*
ViewClip	name	*(Optional; PDF 1.4)* The name of the page boundary to which the contents of a page are to be clipped when viewing the document on the screen. 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.*
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 name of the page boundary to which the contents of a page are to be clipped 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.*
PrintScaling	name	*(Optional; PDF 1.6)* The page scaling option to be selected when a print dialog is displayed for this document. Valid values are None, which indicates that the print dialog should reflect no page scaling, and AppDefault, which indicates that applications should use the current print scaling. If this entry has an unrecognized value, applications should use the current print scaling. Default value: AppDefault. ***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 the ability to pick the input tray by size.* Default value: as defined by the PDF viewer application

KEY	TYPE	VALUE
PrintPageRange	array	(*Optional; PDF 1.7*) The page numbers used to initialize the print dialog box when the file is printed. The first page of the PDF file is denoted by 1. Each pair consists of the first and last pages in the sub-range. An odd number of integers causes this entry to be ignored. Negative numbers cause the entire array to be ignored. Default value: as defined by PDF viewer application
NumCopies	integer	(*Optional; PDF 1.7*) The number of copies to be printed when the print dialog is opened for this file. Supported values are the integers 2 through 5. Values outside this range are ignored. Default value: as defined by PDF viewer application, but typically 1

8.2 Document-Level Navigation

The features described in this section allow a PDF viewer application to present the user with an interactive, global overview of a document in either of two forms:

- 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
- The location of the document window on that page
- The magnification (zoom) factor to use when displaying the page

Destinations may be associated with outline items (see Section 8.2.2, "Document Outline"), annotations ("Link Annotations" on page 622), or actions ("Go-To Actions” on page 654 and “Remote Go-To Actions” on page 655). In each case, the destination specifies the view of the document to be presented when the outline item or annotation is opened or the action is performed. In addition, the optional **OpenAction** entry in a document’s catalog (Section 3.6.1, “Document Catalog”) may specify a destination to be displayed when the document is opened. A destination may be specified either explicitly by an array of parameters defining its properties or indirectly by name.

Explicit Destinations

Table 8.2 shows the allowed syntactic forms for specifying a destination explicitly in a PDF file. In each case, *page* is an indirect reference to a page object. All coordinate values (*left*, *right*, *top*, and *bottom*) are expressed in the default user space coordinate system. The page’s *bounding box* is the smallest rectangle enclosing all of its contents. (If any side of the bounding box lies outside the page’s crop box, the corresponding side of the crop box is used instead; see Section 10.10.1, “Page Boundaries,” for further discussion of the crop box.)

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 the smaller of the two, centering the page within the window in the other dimension.

SYNTAX	**MEANING**
[*page* /FitH *top*]	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 the page within the window. A null value for *top* specifies that the current value of that parameter is to be retained unchanged.
[*page* /FitV *left*]	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 the page within the window. A null value for *left* specifies that the current value of that parameter is to be retained unchanged.
[*page* /FitR *left bottom right top*]	Display the page designated by *page*, with its contents magnified just enough to fit the rectangle specified by the coordinates *left*, *bottom*, *right*, and *top* 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 rectangle within the window in the other dimension. A null value for any of the parameters may result in unpredictable behavior.
[*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.

Named Destinations

Instead of being defined directly with the explicit syntax shown in Table 8.2, a destination may be referred to indirectly by means of a name object *(PDF 1.1)* or a byte string *(PDF 1.2)*. This capability is especially useful when the destination is located in another PDF document. For example, a link to the beginning of Chapter 6 in another document might refer to the destination by a name, such as Chap6.begin, instead of by an explicit page number in the other document. Then, the location of the chapter in the other document could change without invalidating the link. If an annotation or outline item that refers to a named destination has an associated action, such as a remote go-to action (see "Remote Go-To Actions" on page 655) or a thread action ("Thread Actions" on page 661), the destination is in the file specified by the action's **F** entry, if any; if there is no **F** entry, the destination is in the current file.

In PDF 1.1, the correspondence between name objects and destinations is defined by the **Dests** entry in the document catalog (see Section 3.6.1, "Document Catalog"). The value of this entry is a dictionary in which each key is a destination name and the corresponding value is either an array defining the destination, using the syntax shown in Table 8.2, or a dictionary with a **D** entry whose value is such an array. The latter form allows additional attributes to be associated with the destination, as well as enabling a go-to action (see "Go-To Actions" on page 654) to be used as the target of a named destination.

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

A PDF document may optionally display a *document outline* on the screen, allowing the user to navigate interactively from one part of the document to another. The outline consists of a tree-structured hierarchy of *outline items* (sometimes called *bookmarks*), which serve as a visual table of contents to display the document's structure to the user. The user can interactively open and close individual items by clicking them with the mouse. When an item is open, its immediate children in the hierarchy become visible on the screen; each child may in turn be open or closed, selectively revealing or hiding further parts of the hierarchy. When an item is closed, all of its descendants in the hierarchy are hidden. Clicking the text of any visible item *activates* the item, causing the viewer application to jump to a destination or trigger an action associated with the item.

The root of a document's outline hierarchy is an *outline dictionary* specified by the **Outlines** entry in the document catalog (see Section 3.6.1, "Document Catalog"). Table 8.3 shows the contents of this dictionary. Each individual outline item within the hierarchy is defined by an *outline item dictionary* (Table 8.4). The items at each level of the hierarchy form a linked list, chained together through their **Prev** and **Next** entries and accessed through the **First** and **Last** entries in the parent item (or in the outline dictionary in the case of top-level items). When displayed on the screen, the items at a given level appear in the order in which they occur in the linked list. (See also implementation note 74 in Appendix H.)

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

KEY	TYPE	VALUE
Title	text string	*(Required)* The text to be displayed on the screen for this item.
Parent	dictionary	*(Required; must be an indirect reference)* The parent of this item in the outline hierarchy. The parent of a top-level item is the outline dictionary itself.

KEY	TYPE	VALUE
Prev	dictionary	*(Required for all but the first item at each level; must be an indirect reference)* The previous item at this outline level.
Next	dictionary	*(Required for all but the last item at each level; must be an indirect reference)* The next item at this outline level.
First	dictionary	*(Required if the item has any descendants; must be an indirect reference)* The first of this item's immediate children in the outline hierarchy.
Last	dictionary	*(Required if the item has any descendants; must be an indirect reference)* The last of this item's immediate children in the outline hierarchy.
Count	integer	*(Required if the item has any descendants)* If the item is open, the total number of its open descendants at all lower levels of the outline hierarchy. If the item is closed, a negative integer whose absolute value specifies how many descendants would appear if the item were reopened.
Dest	name, byte string, or array	*(Optional; not permitted if an **A** entry is present)* The destination to be displayed when this item is activated (see Section 8.2.1, "Destinations"; see also 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 displaying the outline item's text (see Table 8.5). Default value: 0.

The value of the outline item dictionary's **F** entry *(PDF 1.4)* is an unsigned 32-bit integer containing flags specifying style characteristics for displaying the item. Bit positions within the flag word are numbered from 1 (low-order) to 32 (high-order). Table 8.5 shows the meanings of the flags; all undefined flag bits are reserved and must be set to 0.

TABLE 8.5 Outline item flags

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 and outline item dictionary. See Appendix G for an example of a complete outline hierarchy.

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 screen, allowing the user to navigate to a page by clicking its thumbnail image:

Note: *Thumbnail images are not required, and may be included for some pages and not for others.*

The thumbnail image for a page is an image XObject specified by the **Thumb** entry in the page object (see "Page Objects" on page 144). It has the usual structure for an image dictionary (Section 4.8.4, "Image Dictionaries"), but only the **Width**, **Height**, **ColorSpace**, **BitsPerComponent**, and **Decode** entries are significant; all of the other entries listed in Table 4.39 on page 340 are ignored if present. (If a **Subtype** entry is specified, its value must be **Image**.) The image's color space must be either **DeviceGray** or **DeviceRGB**, or an **Indexed** space based on one of these. Example 8.2 shows a typical thumbnail image definition.

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 sets. There is no requirement that files in a collection have an implicit relationship or even a similarity; however, showing differentiating characteristics of related documents can be helpful for document navigation.

A *collection dictionary* specifies the viewing and organizational characteristics of portable collections. If this dictionary is present in a PDF document, the user interface presents the document as a portable collection. The **EmbeddedFiles** name tree specifies file attachments (see Section 3.10.3, "Embedded File Streams).

When a PDF 1.7-compliant viewer application first opens a PDF document containing a collection, it must display the contents of the initial document, along with a list of the documents present in the **EmbeddedFiles** name tree. The document list must include the additional document information specified by the collection schema. The initial document can be the container PDF or one of the embedded documents.

The page content in the initial document typically contains information that helps the viewer understand what is contained in the collection, such as a title and an introductory paragraph.

The file attachments comprising a collection are located in the **EmbeddedFiles** name tree. All attachments in that tree are in the collection; any attachments not 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 document is the one that contains the collection dictionary.

KEY	TYPE	VALUE
View	name	(*Optional*) The initial view. The following values are valid: **D** The collection view is presented in details mode, with all information in the Schema dictionary presented in a multi-column format. This mode provides the most information to the user. **T** The collection view is presented in tile mode, with each file in the collection denoted by a small icon and a subset of information from the **Schema** dictionary. This mode provides top-level information about the file attachments to the user. **H** The collection view is initially hidden, without preventing the user from obtaining a file list via explicit action. Default value: **D**
Sort	dictionary	(*Optional*) A collection sort dictionary, which specifies the order in which items in the collection should be sorted in the user interface (see Table 8.9 on page 592).

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 dictionary.

A *collection field dictionary* describes the attributes of a particular field in a portable collection, including the type of data stored in the field and the lookup key used to locate the field data in the file specification dictionary. Table 8.8 describes the entries in a collection field dictionary.

TABLE 8.8 Entries in a collection field dictionary

KEY	TYPE	VALUE
Type	name	(*Optional*) The type of PDF object that this dictionary describes; if present, must be CollectionField for a collection field dictionary.
Subtype	name	(*Required*) The subtype of collection field or file-related field that this dictionary describes. This entry identifies the type of data that is stored in the field. The following values identify the types of fields in the collection item or collection subitem dictionary: **S** A text field. The field data is stored as a PDF text string. **D** A date field. The field data is stored as a PDF date string. **N** A number field. The field data is stored as a PDF number. The following values identify the types of file-related fields: **F** The field data is the file name of the embedded file stream, as identified 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 viewer application.
O	integer	(*Optional*) The relative order of the field name in the user interface. Fields are sorted by the PDF viewer application in ascending order.

KEY	TYPE	VALUE
V	boolean	(*Optional*) The initial visibility of the field in the user interface. Default value: true.
E	boolean	(*Optional*) A flag indicating whether the PDF viewer application should provide support for editing the field value. Default value: false.

A *collection sort dictionary* identifies the fields that are used to sort items in the collection. The type of sorting depends on the type of data:

- Text strings are ordered lexically from smaller to larger, if ascending order is specified.
- Numbers are ordered numerically from smaller to larger, if ascending order is specified.
- Dates are ordered from oldest to newest, if ascending order is specified.

Table 8.9 describes the entries in a collection sort dictionary.

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	boolean or array	(*Optional*) Specifies whether the items in the collection are sorted in ascending order. If the array form is used, each element of the array is a boolean value that specifies whether the entry at the same index in the **S** array is sorted in ascending order. Default value: true.

Example 8.3 shows a collection dictionary representing an email in-box, where each item in the collection is an email message. The actual email messages are contained in file specification dictionaries. The organizational data associated with each email is described in a collection schema dictionary. Most actual organizational data (from, to, date, and subject) is provided in a collection item dictionary, but the size data comes from the embedded file parameter dictionary.

Example 8.3

```
/Collection <<
    /Type /Collection
    /Schema <<
        /Type /CollectionSchema
        /from << /Subtype /S /N (From) /O 1 /V true /E false>>
        /to << /Subtype /S /N (To) /O 2 /V true /E false >>
        /date << /Subtype /D /N (Date received) /O 3 /V true /E false >>
        /subject << /Subtype /S /N (Subject) /O 4 /V true /E false >>
        /size << /Subtype /Size /N (Size) /O 5 /V true /E false >>
        >>
    /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:)
        /D (Let's have lunch on Friday!)
    >>
    /date (D:20050621094703-07'00')
>>
```

8.3 Page-Level Navigation

This section describes PDF facilities that enable the user to navigate from page to page within a document:

- *Page labels* for numbering or otherwise identifying individual pages (see Section 8.3.1)
- *Article threads*, which chain together items of content within the document that are logically connected but not physically sequential (see Section 8.3.2)
- *Presentations* that display the document in the form of a slide show, advancing from one page to the next either automatically or under user control (see Section 8.3.3)

For another important form of page-level navigation, see "Link Annotations" on 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, optionally preceded by a *label prefix* denoting the range itself. For example, the pages in an appendix might be labeled with decimal numeric portions prefixed with the string A–; the resulting page labels would be A–1, A–2, and so on.

A document's labeling ranges are defined by the **PageLabels** entry in the document catalog (see Section 3.6.1, "Document Catalog"). The value of this entry is a number tree (Section 3.8.6, “Number Trees”), each of whose keys is the page index of the first page in a labeling range. The corresponding value is a *page label dictionary* defining the labeling characteristics for the pages in that range. The tree must include a value for page index 0. Table 8.10 shows the contents of a page label dictionary. (See implementation note 76 in Appendix H.)

Example 8.5 shows a document with pages labeled

i, ii, iii, iv, 1, 2, 3, A–8, A–9, …

Example 8.5

```
1 0 obj
   << /Type /Catalog
      /PageLabels << /Nums [ 0 << /S /r >>       % A number tree containing
                             4 << /S /D >>       %    three page label dictionaries
                             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) a Lowercase letters (a to z for the first 26 pages, aa to zz for the next 26, and so on) There is no default numbering style; if no **S** entry is present, page labels consist solely of a label prefix with no numeric portion. For example, if the **P** entry (below) specifies the label prefix Contents, each page is simply labeled Contents with no page number. (If the **P** entry is also missing or empty, the page label is an empty string.)

KEY	TYPE	VALUE
P	text string	*(Optional)* The label prefix for page labels in this range.
St	integer	*(Optional)* The value of the numeric portion for the first page label in the range. Subsequent pages are numbered sequentially from this value, which must be greater than or equal to 1. Default value: 1.

8.3.2 Articles

Some types of documents may contain sequences of content items that are logically connected but not physically sequential. For example, a news story may begin on the first page of a newsletter and run over onto one or more nonconsecutive interior pages. To represent such sequences of physically discontiguous but logically related items, a PDF document may define one or more *articles (PDF 1.1)*. The sequential flow of an article is defined by an *article thread*; the individual content items that make up the article are called *beads* on the thread. PDF viewer applications can provide navigation facilities to allow the user to follow 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	dictionary	*(Required; must be an indirect reference)* The first bead in the thread.
I	dictionary	*(Optional)* A thread information dictionary containing information about the thread, such as its title, author, and creation date. The contents of this dictionary are similar to those of the document information dictionary (see Section 10.2.1, "Document Information Dictionary").

TABLE 8.12 Entries in a bead dictionary

KEY	TYPE	VALUE
Type	name	*(Optional)* The type of PDF object that this dictionary describes; if present, must be **Bead** for a bead dictionary.
T	dictionary	*(Required for the first bead of a thread; optional for all others; must be an indirect reference)* The thread to which this bead belongs. ***Note:** In PDF 1.1, this entry is permitted only for the first bead of a thread. In PDF 1.2 and higher, it is permitted for any bead but required only for the first.*
N	dictionary	*(Required; must be an indirect reference)* The next bead in the thread. In the last bead, this entry points to the first.
V	dictionary	*(Required; must be an indirect reference)* The previous bead in the thread. In the first bead, this entry points to the last.
P	dictionary	*(Required; must be an indirect reference)* The page object representing the page on 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]
   >>
endobj
```

```
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.)

The **Dur** entry in the page object specifies the page's *display duration* (also called its *advance timing*): the maximum length of time, in seconds, that the page is displayed before the presentation automatically advances to the next page. (The user can advance the page manually before the specified time has expired.) If no **Dur** entry is specified in the page object, the page does not advance automatically.

TABLE 8.13 Entries in a transition dictionary

KEY	TYPE	VALUE	
Type	name	*(Optional)* The type of PDF object that this dictionary describes; if present, must be **Trans** for a transition dictionary.	
S	name	*(Optional)* The *transition style* to use when moving to this page from another during a presentation. Default value: R.	
		Split	Two lines sweep across the screen, revealing the new page. The lines may be either horizontal or vertical and may move inward from the edges of the page or outward from the center, as specified by the **Dm** and **M** entries, respectively.
		Blinds	Multiple lines, evenly spaced across the screen, synchronously sweep in the same direction to reveal the new page. The lines may be either horizontal or vertical, as specified by the **Dm** entry. Horizontal lines move downward; vertical lines move to the right.
		Box	A rectangular box sweeps inward from the edges of the page or outward 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 by **Di**, covering the old page.
		Uncover	*(PDF 1.5)* The old page slides off the screen in the direction specified by **Di**, uncovering the new page in the direction specified by **Di**.
		Fade	*(PDF 1.5)* The new page gradually becomes visible through the old one.

KEY	TYPE	VALUE
D	number	*(Optional)* The duration of the transition effect, in seconds. Default value: 1.
Dm	name	*(Optional; Split and Blinds transition styles only)* The dimension in which the specified transition effect occurs: H Horizontal V Vertical Default value: H.
M	name	*(Optional; Split, Box and Fly transition styles only)* The direction of motion for the specified transition effect: I Inward from the edges of the page O Outward from the center of the page Default value: I.
Di	number or name	*(Optional; Wipe, Glitter, Fly, Cover, Uncover and Push transition styles only)* The direction in which the specified transition effect moves, expressed in degrees counterclockwise starting from a left-to-right direction. (This differs from the page object's **Rotate** entry, 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 Default: 1.0.
B	boolean	(*Optional; PDF 1.5; Fly transition style only*) If **true**, the area to be flown in is rectangular and opaque. Default: **false**.

Figure 8.1 illustrates the relationship between transition duration (**D** in the transition dictionary) and display duration (**Dur** in the page object). Note that the tran-

sition duration specified for a page (page 2 in the figure) governs the transition *to* that page from another page; the transition *from* the page is governed by the next page's transition duration.

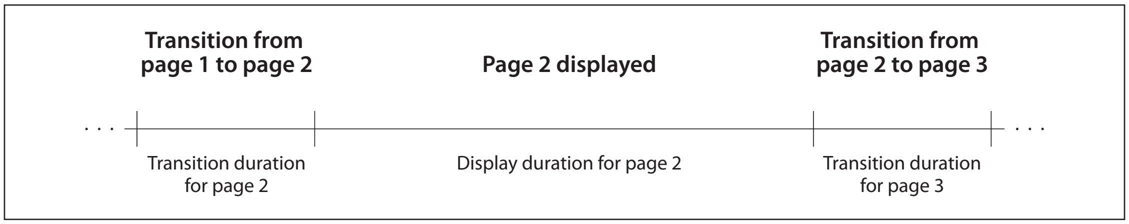

Transition from page 1 to page 2
Page 2 displayed
Transition from page 2 to page 3
Transition duration for page 2
Display duration for page 2
Transition duration for page 3

FIGURE 8.1 *Presentation timing*

Example 8.7 shows the presentation parameters for a page to be displayed for 5 seconds. Before the page is displayed, there is a 3.5-second transition in which two vertical lines sweep outward from the center to the edges of the page.

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

Sub-page navigation (PDF 1.5) allows navigating not only between pages but also between different states of the same page. For example, a single page in a PDF presentation could have a series of bullet points that could be individually turned on and off. In such an example, the bullets would be represented by optional content (see Section 4.10, "Optional Content"), and each state of the page would be represented as a *navigation node.*

***Note:** Viewer applications should save the state of optional content groups when a user enters presentation mode and restore it when presentation mode ends. This ensures, for example, that transient changes to bullets do not affect the printing of the document.*

A navigation node dictionary (see Table 8.14) specifies actions to execute when the user makes a navigation request; for example, by pressing an arrow key. The navigation nodes on a page form a doubly linked list by means of their **Next** and **Prev** entries. The primary node on a page is determined by the optional **PresSteps** entry in a page dictionary (see Table 3.27).

***Note:** It is recommended that a viewer application respect navigation nodes only when in presentation mode (see Section 8.3.3, "Presentations").*

TABLE 8.14 Entries in a navigation node dictionary

KEY	TYPE	VALUE
Type	name	*(Optional)* The type of PDF object that this dictionary describes; must be **NavNode** 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:

1. The sequence of actions specified by **NA** (if present) is executed.

   ***Note:** If **NA** specifies an action that navigates to another page, the actions described below for navigating to another page take place, and **Next** should not be present.*

2. The node specified by **Next** (if present) becomes the new current navigation node.

Similarly, if there is a request to navigate backward and there is a current navigation node, the following occurs:

1. The sequence of actions specified by **PA** (if present) is executed.

   Note: *If* ***PA*** *specifies an action that navigates to another page, the actions described below for navigating to another page take place, and* ***Prev*** *should not be present.*

2. The node specified by **Prev** (if present) becomes the new current navigation node.

When navigating between nodes, it is possible to specify transition effects. These effects are similar to the page transitions specified in the previous section. However, they use a different mechanism; see "Transition Actions" on page 670.

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:

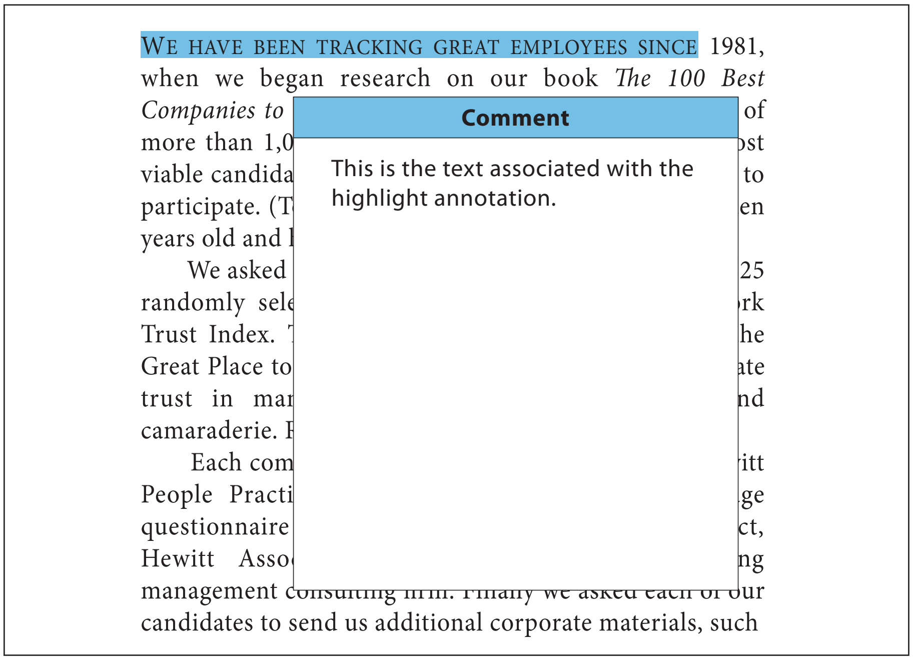
1. The navigation node represented by **PresSteps** becomes the current node.
2. If the navigation request was forward, or if the navigation request was for random access (such as by clicking on a link), the actions specified by **NA** are executed and the node specified by **Next** becomes the new current 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.
3. The viewer application makes the new page the current page and displays it. Any page transitions specified by the **Trans** entry of the page dictionary are performed.

8.4 Annotations

An *annotation* associates an object such as a note, sound, or movie with a location on a page of a PDF document, or provides a way to interact with the user by means of the mouse and keyboard. PDF includes a wide variety of standard annotation types, described in detail in Section 8.4.5, “Annotation Types.”

Many of the standard annotation types may be displayed in either the *open* or the *closed* state. When closed, they appear on the page in some distinctive form, such as an icon, a box, or a rubber stamp, depending on the specific annotation type. When the user *activates* the annotation by clicking it, it exhibits its associated object, such as by opening a pop-up window displaying a text note (Figure 8.2) or by playing a sound or a movie.



WE HAVE BEEN TRACKING GREAT EMPLOYEES SINCE 1981,
when we began research on our book *The 100 Best*
Companies to of
more than 1,0 ost
viable candida to
participate. (T en
years old and l
We asked 25
randomly sele rk
Trust Index. T he
Great Place to ate
trust in mar nd
camaraderie. F
Each com itt
People Practi ge
questionnaire ct,
Hewitt Asso ng
management consulting firm. Finally we asked each of our
candidates to send us additional corporate materials, such

Comment

This is the text associated with the highlight annotation.

FIGURE 8.2 *Open annotation*

Viewer applications may permit the user to navigate through the annotations on a page by using the keyboard (in particular, the tab key); see implementation note 77 in Appendix H. Beginning with PDF 1.5, PDF producers may make the navigation order explicit with the optional **Tabs** entry in a page object (see Table 3.27). The following are the possible values for this entry:

- R (row order): Annotations are visited in rows running horizontally across the page. The direction within a row is determined by the **Direction** entry in the viewer preferences dictionary (see Section 8.1, "Viewer Preferences"). The first annotation visited is the first annotation in the topmost row. When the end of a row is encountered, the first annotation in the next row is visited.
- C (column order): Annotations are visited in columns running vertically up and down the page. Columns are ordered by the **Direction** entry in the viewer preferences dictionary (see Section 8.1, "Viewer Preferences"). The first annotation visited is the one at the top of the first column. When the end of a column is encountered, the first annotation in the next column is visited.
- S (structure order): Annotations are visited in the order in which they appear in the structure tree (see Section 10.6, "Logical Structure"). The order for annotations that are not included in the structure tree is application-dependent.

***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.

***Note:** A given annotation dictionary may be referenced from the **Annots** array of only one page. Attempting to share an annotation dictionary among multiple pages produces unpredictable behavior. This requirement applies only to the annotation dictionary itself, not to subsidiary objects, which can be shared among multiple annotations without causing any difficulty.*

TABLE 8.15 Entries common to all annotation dictionaries

KEY	TYPE	VALUE
Type	name	*(Optional)* The type of PDF object that this dictionary describes; if present, must be **Annot** for an annotation dictionary.
Subtype	name	*(Required)* The type of annotation that this dictionary describes; see Table 8.20 on page 615 for specific values.
Rect	rectangle	*(Required)* The *annotation rectangle*, defining the location of the annotation on the page in default user space units.
Contents	text string	*(Optional)* Text to be displayed for the annotation or, if this type of annotation does not display text, an alternate description of the annotation's contents in human-readable form. In either case, this text is 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"). See Section 8.4.5, "Annotation Types" for more details on the meaning of this entry for each annotation 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 presented visually on the page (see Section 8.4.4, "Appearance Streams" and also implementation note 79 in Appendix H). Individual annotation handlers may ignore this entry and provide their own appearances.

KEY	TYPE	VALUE
AS	name	*(Required if the appearance dictionary **AP** contains one or more subdictionaries; PDF 1.2)* The annotation's *appearance state*, which selects the applicable appearance stream from an appearance subdictionary (see Section 8.4.4, "Appearance Streams" and also implementation note 79 in Appendix H).
Border	array	*(Optional)* An array specifying the characteristics of the annotation's border. The border is specified as a rounded rectangle. In PDF 1.0, the array consists of three numbers defining the horizontal corner radius, vertical corner radius, and border width, all in default user space units. If the corner radii are 0, the border has square (not rounded) corners; if the border width is 0, no border is drawn. (See implementation note 81 in Appendix H.) In PDF 1.1, the array may have a fourth element, an optional *dash array* defining a pattern of dashes and gaps to be used in drawing the border. 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). For example, a **Border** value of `[0 0 1 [3 2]]` specifies a border 1 unit wide, with square corners, 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: 0 No color; transparent 1 **DeviceGray** 3 **DeviceRGB** 4 **DeviceCMYK**

KEY	TYPE	VALUE
StructParent	integer	*(Required if the annotation is a structural content item; PDF 1.3)* The integer key of the annotation's entry in the structural parent tree (see "Finding Structure Elements from Content Items" on page 868).
OC	dictionary	*(Optional; PDF 1.5)* An optional content group or optional content membership dictionary (see Section 4.10, "Optional Content") specifying the optional content properties for the annotation. Before the annotation is drawn, its visibility is determined based on this entry as well as the annotation flags specified in the **F** entry (see Section 8.4.2, "Annotation Flags"). If it is determined to be invisible, the annotation is skipped, as if it were not in the document.

8.4.2 Annotation Flags

The value of the annotation dictionary's **F** entry is an unsigned 32-bit integer containing flags specifying various characteristics of the annotation. Bit positions within the flag word are numbered from 1 (low-order) to 32 (high-order). Table 8.16 shows the meanings of the flags; all undefined flag bits are reserved and must 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	Print	*(PDF 1.2)* If set, print the annotation when the page is printed. If clear, never print the annotation, regardless of whether it is displayed on the screen. This can be useful, for example, for annotations representing interactive pushbuttons, which would serve no meaningful purpose on the printed page. (See implementation note 83 in Appendix H.)

BIT POSITION	NAME	MEANING
4	NoZoom	*(PDF 1.3)* If set, do not scale the annotation's appearance to match the magnification of the page. The location of the annotation on the page (defined by the upper-left corner of its annotation rectangle) remains fixed, regardless of the page magnification. See below for further discussion.
5	NoRotate	*(PDF 1.3)* If set, do not rotate the annotation's appearance to match the rotation of the page. The upper-left corner of the annotation rectangle remains in a fixed location on the page, regardless of the page rotation. See below for further discussion.
6	NoView	*(PDF 1.3)* If set, do not display the annotation on the screen or allow it to interact with the user. The annotation may be printed (depending on the setting of the Print flag) but should be considered hidden for purposes of on-screen display and user interaction.
7	ReadOnly	*(PDF 1.3)* If set, do not allow the annotation to interact with the user. The annotation may be displayed or printed (depending on the settings of the NoView and Print flags) but should not respond to mouse clicks or change its 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 the screen and is unaffected by the magnification level at which the page itself is displayed. Similarly, if the NoRotate flag is set, the annotation retains its original orientation on the screen when the page is rotated (by changing the **Rotate** entry in the page object; see "Page Objects" on page 144).

In either case, the annotation's position is determined by the coordinates of the upper-left corner of its annotation rectangle, as defined by the **Rect** entry in the annotation dictionary and interpreted in the default user space of the page. When the default user space is scaled or rotated, the positions of the other three corners of the annotation rectangle are different in the altered user space than they were in the original user space. The viewer application performs this alteration automatically. However, it does not actually change the annotation's **Rect** entry, which continues to describe the annotation's relationship with the unscaled, unrotated user space.

For example, Figure 8.3 shows how an annotation whose NoRotate flag is set remains upright when the page it is on is rotated 90 degrees clockwise. The upper-left corner of the annotation remains at the same point in default user space; the annotation pivots around that point.

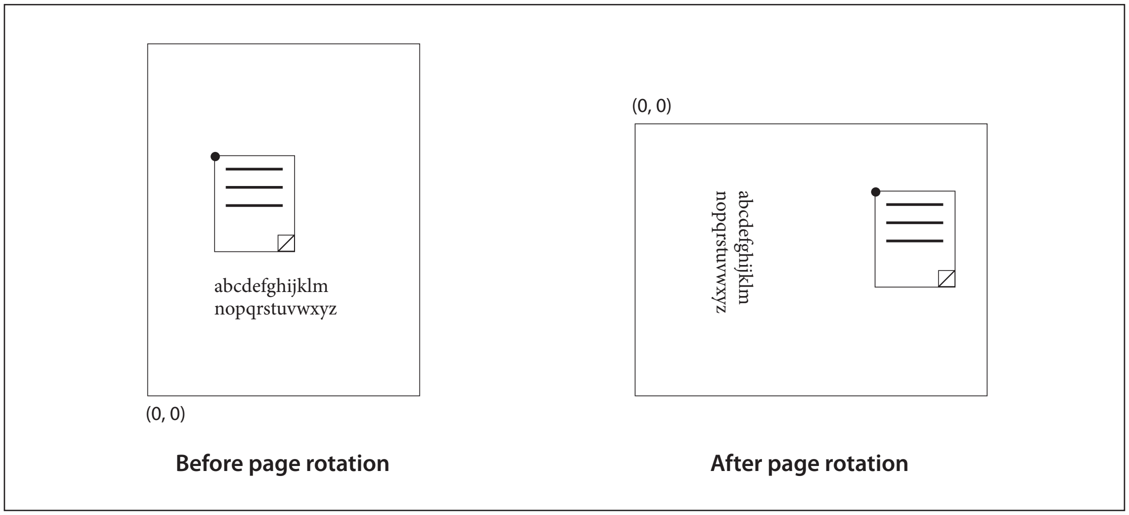


FIGURE 8.3 *Coordinate adjustment with the NoRotate flag*

8.4.3 Border Styles

An annotation may optionally be surrounded by a border when displayed or printed. If present, the border is drawn completely inside the annotation rectangle. In PDF 1.1, the characteristics of the border are specified by the **Border** entry in the annotation dictionary (see Table 8.15 on page 606). Beginning with

PDF 1.2, some types of annotations may instead specify their border characteristics in a *border style dictionary* designated by the annotation's **BS** entry. Such dictionaries are also used to specify the width and dash pattern for the lines drawn by line, square, circle, and ink annotations. Table 8.17 summarizes the contents of the border style dictionary. If neither the **Border** nor the **BS** entry is present, the border is drawn as a solid line with a width of 1 point.

TABLE 8.17 Entries in a border style dictionary

KEY	TYPE	VALUE
Type	name	*(Optional)* The type of PDF object that this dictionary describes; if present, must be **Border** for a border style dictionary.
W	number	*(Optional)* The border width in points. If this value is 0, no border is drawn. Default value: 1.
S	name	*(Optional)* The border style: S (Solid) A solid rectangle surrounding the annotation. 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].

Beginning with PDF 1.5, some annotations (square, circle, and polygon) can have a **BE** entry, which is a *border effect dictionary* that specifies an effect to be applied to the border of the annotations. Beginning with PDF 1.6, the free text annotation can also have a **BE** entry. Table 8.18 describes the entries in a border effect dictionary.

TABLE 8.18 Entries in a border effect dictionary

KEY	TYPE	VALUE
S	name	*(Optional)* A name representing the border effect to apply. Possible values are: S No effect: the border is as described by the annotation dictionary's **BS** entry. C The border should appear "cloudy". The width and dash array specified by **BS** are honored. Default value: S.
I	number	*(Optional; valid only if the value of* ***S*** *is C)* A number describing the intensity of the effect. Suggested values range from 0 to 2. Default value: 0.

8.4.4 Appearance Streams

Beginning with PDF 1.2, an annotation can specify one or more *appearance streams* as an alternative to the simple border and color characteristics available 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 transformed appearance box to the corresponding corners of the annotation's rectangle.

3. **Matrix** is concatenated with A to form a matrix AA that maps from the appearance's coordinate system to the annotation's rectangle in default user space:

   $AA = A \times \mathbf{Matrix}$

The annotation may be further scaled and rotated if either the NoZoom or NoRotate flag is set (see Section 8.4.2, "Annotation Flags"). Any transformation applied to the annotation as a whole is also applied to the appearance within it.

In PDF 1.4, an annotation appearance can include transparency. If the appearance's stream dictionary does not contain a **Group** entry, it is treated as a non-isolated, non-knockout transparency group. Otherwise, the isolated and knockout values specified in the group dictionary (see Section 7.5.5, "Transparency Group XObjects") are used.

The transparency group is composited with a backdrop consisting of the page content along with any previously painted annotations, using a blend mode of **Normal**, an alpha constant of 1.0, and a soft mask of **None**. (See implementation note 87 in Appendix H.)

Note: *If a transparent annotation appearance is painted over an annotation that is drawn without using an appearance stream, the effect is implementation-dependent. This is because such annotations are sometimes drawn by means that do not 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.*

The normal, rollover, and down appearances are defined in an *appearance dictionary*, which in turn is the value of the **AP** entry in the annotation dictionary (see Table 8.15 on page 606). Table 8.19 shows the contents of the appearance dictionary.

TABLE 8.19 Entries in an appearance dictionary

KEY	TYPE	VALUE
N	stream or dictionary	*(Required)* The annotation's normal appearance.
R	stream or dictionary	*(Optional)* The annotation's rollover appearance. Default value: the value of the **N** entry.
D	stream or dictionary	*(Optional)* The annotation's down appearance. Default value: the value of the **N** entry.

Each entry in the appearance dictionary may contain either a single appearance stream or an *appearance subdictionary*. In the latter case, the subdictionary defines multiple appearance streams corresponding to different *appearance states* of the annotation.

For example, an annotation representing an interactive check box might have two appearance states named On and Off. Its appearance dictionary might be defined 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).

***Note:** Some of the standard PDF annotation types, such as movie annotations—as well as all custom annotation types defined by third parties—are implemented through plug-in extensions. If the plug-in for a particular annotation type is not available, PDF viewer applications should display the annotation with its normal (**N**) appearance. Viewer applications should also attempt to provide reasonable be-*

havior (such as displaying nothing) if an annotation's **AS** *entry designates an appearance state for which no appearance is defined in the appearance dictionary.*

For convenience in managing appearance streams that are used repeatedly, the **AP** entry in a PDF document's name dictionary (see Section 3.6.3, "Name Dictionary") can contain a name tree mapping name strings to appearance streams. The name strings have no standard meanings; no PDF objects refer to appearance streams by name.

8.4.5 Annotation Types

PDF supports the standard annotation types listed in Table 8.20. The following sections describe each of these types in detail. Plug-in extensions may add new annotation types, and further standard types may be added in the future. (See implementation note 88 in Appendix H.)

The values in the first column of Table 8.20 represent the value of the annotation dictionary's **Subtype** entry. The third column indicates whether the annotation is 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
Polygon	*(PDF 1.5)* Polygon annotation	Yes	"Polygon and Polyline Annotations" on page 632
PolyLine	*(PDF 1.5)* Polyline annotation	Yes	"Polygon and Polyline Annotations" on page 632

ANNOTATION TYPE	**DESCRIPTION**	**MARKUP?**	**DISCUSSED IN SECTION**
Highlight	*(PDF 1.3)* Highlight annotation	Yes	"Text Markup Annotations" on page 633
Underline	*(PDF 1.3)* Underline annotation	Yes	"Text Markup Annotations" on page 633
Squiggly	*(PDF 1.4)* Squiggly-underline annotation	Yes	"Text Markup Annotations" on page 633
StrikeOut	*(PDF 1.3)* Strikeout annotation	Yes	"Text Markup Annotations" on page 633
Stamp	*(PDF 1.3)* Rubber stamp annotation	Yes	"Rubber Stamp Annotations" on page 635
Caret	*(PDF 1.5)* Caret annotation	Yes	"Caret Annotations" on page 634
Ink	*(PDF 1.3)* Ink annotation	Yes	"Ink Annotations" on page 636
Popup	*(PDF 1.3)* Pop-up annotation	No	"Pop-up Annotations" on page 637
FileAttachment	*(PDF 1.3)* File attachment annotation	Yes	"File Attachment Annotations" on page 637
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 annotation's **Contents** entry varies by annotation type. Typically, it is the text to be displayed for the annotation or, if the annotation does not display text, an alternate description of the annotation's contents in human-readable form. In either case, the **Contents** entry is 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").

Many annotation types are defined as *markup annotations* because they are used primarily to mark up PDF documents (see Table 8.20). These annotations have text that appears as part of the annotation and may be displayed in other ways by a viewer application, such as in a Comments pane.

Markup annotations can be divided into the following groups:

- Free text annotations display text directly on the page. The annotation's **Contents** entry specifies the displayed text.
- Most other markup annotations have an associated pop-up window that may contain text. The annotation's **Contents** entry specifies the text to be displayed when the pop-up window is opened. These include text, line, square, circle, polygon, polyline, highlight, underline, squiggly-underline, strikeout, rubber stamp, caret, ink, and file attachment annotations.
- Sound annotations do not have a pop-up window but may also have associated 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 purposes (see Section 10.8.2, "Alternate Descriptions").

Table 8.21 lists entries that apply to all markup annotations.

TABLE 8.21 Additional entries specific to markup annotations

KEY	TYPE	VALUE
T	text string	*(Optional; PDF 1.1)* The text label to be displayed in the title bar of the annotation's pop-up window when open and active. By convention, this entry identifies the user who added the annotation.
Popup	dictionary	*(Optional; PDF 1.3)* An indirect reference to a pop-up annotation for entering or editing the text associated with this annotation.
CA	number	*(Optional; PDF 1.4)* The constant opacity value to be used in painting the annotation (see Sections 7.1, "Overview of Transparency," and 7.2.6, "Shape and Opacity Computations"). This value applies to all visible elements of the annotation in its closed state (including its background and border) but not to the pop-up window that appears when the annotation is opened. The specified value is not used if the annotation has an appearance stream (see Section 8.4.4, "Appearance Streams"); in that case, the appearance stream must specify any transparency. (However, if the viewer regenerates the annotation's appearance stream, it may incorporate the **CA** value into the stream's content.) 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. If this entry is present in an FDF file (see Section 8.6.6, "Forms Data Format"), its type is not a dictionary but a text string containing the contents of the **NM** entry of the annotation being replied to, to allow for a situation where the annotation being replied to is not in the same FDF file.

KEY	TYPE	VALUE
Subj	text string	*(Optional; PDF 1.5)* Text representing a short description of the subject being addressed by the annotation.
RT	name	*(Optional; meaningful only if* **IRT** *is present; PDF 1.6)* A name specifying the relationship (the "reply type") between this annotation and one specified by **IRT**. Valid values are: **R** The annotation is considered a reply to the annotation specified by **IRT**. Viewer applications should not display replies to an annotation individually but together in the form of threaded comments. **Group** The annotation is grouped with the annotation specified by **IRT**; see discussion below. Default value: **R**.
IT	name	*(Optional; PDF 1.6)* A name describing the *intent* of the markup annotation. Intents allow viewer applications to distinguish between different uses and behaviors of a single markup annotation type. If this entry is not present or its value is the same as the annotation type, the annotation has no explicit intent and should 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*, which must not have an **IRT** entry, and one or more *subordinate annotations*, which must have an **IRT** entry that refers to the primary annotation and an **RT** entry whose value is **Group**.

Some entries in the primary annotation are treated as "group attributes" that should apply to the group as a whole; the corresponding entries in the subordinate annotations should be ignored. These entries are **Contents** (or **RC** and **DS**), **M**, **C**, **T**, **Popup**, **CreationDate**, **Subj**, and **Open**. Operations that manipulate any annotation in a group, such as movement, cut, and copy, should be treated by viewer applications as acting on the entire group.

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

Annotation States

Beginning with PDF 1.5, annotations may have an author-specific *state* associated with them. The state is not specified in the annotation itself but in a separate text annotation that refers to the original annotation by means of its **IRT** ("in reply to") entry (see Table 8.24). States are grouped into a number of *state models*, as shown in Table 8.22.

TABLE 8.22 Annotation states

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:

- The **T** entry (see Table 8.21) specifies the user.
- The **IRT** entry (see Table 8.24)refers to the original annotation.
- **State** and **StateModel** (see Table 8.23) update the state of the original annotation for the specified user.

Additional state changes are made by adding text annotations in reply to the previous reply for a given user.

Text Annotations

A *text annotation* represents a "sticky note" attached to a point in the PDF document. When closed, the annotation appears as an icon; when open, it displays a pop-up window containing the text of the note in a font and size chosen by the viewer application. Text annotations do not scale and rotate with the page; they behave as if the NoZoom and NoRotate annotation flags (see Table 8.16 on page 608) were always set. Table 8.23 shows the annotation dictionary entries specific to this type of annotation.

TABLE 8.23 Additional entries specific to a text annotation

KEY	TYPE	VALUE
Subtype	name	*(Required)* The type of annotation that this dictionary describes; must be **Text** 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. Default: "Unmarked" if **StateModel** is "Marked"; "None" if **StateModel** is "Review".
StateModel	text string	*(Required if* ***State*** *is present, otherwise optional; PDF 1.5)* The state model corresponding to **State**; see "Annotation States," above.

Example 8.8 shows the definition of a text annotation.

Example 8.8

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

Link Annotations

A *link annotation* represents either a hypertext link to a destination elsewhere in the document (see Section 8.2.1, "Destinations") or an action to be performed (Section 8.5, "Actions"). Table 8.24 shows the annotation dictionary entries specific 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. P (Push) Display the annotation as if it were being pushed below the surface of the page; see implementation note 90 in Appendix H. Default value: I. ***Note:** In PDF 1.1, highlighting is always done by inverting colors inside the annotation rectangle.*

KEY	TYPE	VALUE
PA	dictionary	*(Optional; PDF 1.3)* A URI action (see “URI Actions” on page 662) formerly associated with this annotation. When Web Capture (Section 10.9, “Web Capture”) changes an annotation from a URI to a go-to action (“Go-To Actions” on page 654), it uses this entry to save the data from the original URI action so that it can be changed back in case the target page for the go-to action is subsequently deleted.
QuadPoints	array	*(Optional; PDF 1.6)* An array of $8 \times n$ numbers specifying the coordinates of n quadrilaterals in default user space that comprise the region in which the link should be activated. The coordinates for each quadrilateral are given in the order $x_1\ y_1\ x_2\ y_2\ x_3\ y_3\ x_4\ y_4$ specifying the four vertices of the quadrilateral in counterclockwise order. For orientation purposes, such as when applying an underline border style, the bottom of a quadrilateral is the line formed by (x_1, y_1) and (x_2, y_2). If this entry is not present or the viewer application does not recognize it, the region specified by the **Rect** entry should be used. **QuadPoints** should be ignored if any coordinate in the array lies outside the region specified by **Rect**.

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

A *free text annotation (PDF 1.3)* displays text directly on the page. Unlike an ordinary text annotation (see “Text Annotations” on page 621), a free text annotation has no open or closed state; instead of being displayed in a pop-up window, the text is always visible. Table 8.25 shows the annotation dictionary entries specific to this type of annotation. “Variable Text” on page 677 describes the process of using these entries to generate the appearance of the text in these annotations.

TABLE 8.25 Additional entries specific to a free text annotation

KEY	TYPE	VALUE
Subtype	name	*(Required)* The type of annotation that this dictionary describes; must be **FreeText** for a free text annotation.
DA	string	*(Required)* The default appearance string to be used in formatting the text (see "Variable Text" on page 677). ***Note:*** *The annotation dictionary's* ***AP*** *entry, if present, takes precedence over the* ***DA*** *entry; see Table 8.15 on page 606 and Section 8.4.4, "Appearance Streams."*
Q	integer	*(Optional; PDF 1.4)* A code specifying the form of *quadding* (justification) to be used in displaying the annotation's text: 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 conjunction with the border style dictionary specified by the **BS** entry.

KEY	TYPE	VALUE
RD	rectangle	*(Optional; PDF 1.6)* A set of four numbers describing the numerical differences between two rectangles: the **Rect** entry of the annotation and a rectangle contained within that rectangle. The inner rectangle is where the annotation's text should be displayed. Any border styles and/or border effects specified by **BS** and **BE** entries, respectively, are applied to the border of the inner rectangle. The four numbers correspond to the differences in default user space between the left, top, right, and bottom coordinates of **Rect** and those of the inner rectangle, respectively. Each value must be greater than or equal to 0. The sum of the top and bottom differences must be less than the height of **Rect**, and the sum of the left and right differences must be less than the width of **Rect**.
BS	dictionary	*(Optional; PDF 1.6)* A border style dictionary (see Table 8.17 on page 611) specifying the line width and dash pattern to be used in drawing the annotation's border. ***Note:** The annotation dictionary's **AP** entry, if present, takes precedence over the **InkList** and **BS** entries; see Table 8.15 on page 606 and Section 8.4.4, "Appearance 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].

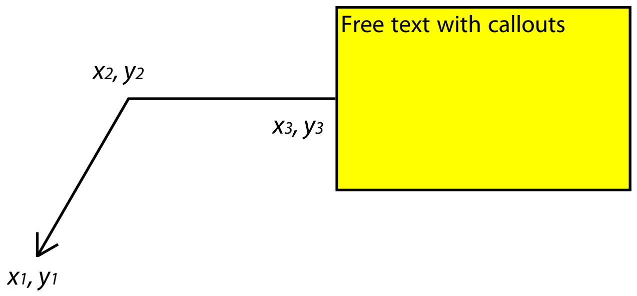


FIGURE 8.4 *Free text annotation with callout*

Line Annotations

A *line annotation (PDF 1.3)* displays a single straight line on the page. When opened, it displays a pop-up window containing the text of the associated note. Table 8.26 shows the annotation dictionary entries specific to this type of annotation.

TABLE 8.26 Additional entries specific to a line annotation

KEY	TYPE	VALUE
Subtype	name	*(Required)* The type of annotation that this dictionary describes; must be **Line** for a line annotation.
L	array	*(Required)* An array of four numbers, [x_1 y_1 x_2 y_2], specifying the starting and ending coordinates of the line in default user space. ***Note:** If the **LL** entry is present, this value represents the endpoints of the leader lines rather than the endpoints of the line itself; see Figure 8.5.*
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** 4 **DeviceCMYK**

KEY	TYPE	VALUE
LL	number	*(Required if **LLE** is present, otherwise optional; PDF 1.6)* The length of *leader lines* in default user space that extend from each endpoint of the line perpendicular to the line itself, as shown in Figure 8.5. A positive value means that the leader lines appear in the direction that is clockwise when traversing the line from its starting point to its ending point (as specified by **L**); a negative value indicates the opposite direction. Default value: 0 (no leader lines).
LLE	number	*(Optional; PDF 1.6)* A non-negative number representing the length of *leader line extensions* that extend from the line proper 180 degrees from the leader lines, as shown in Figure 8.5. Default value: 0 (no leader line extensions).
Cap	boolean	*(Optional; PDF 1.6)* If **true**, the text specified by the **Contents** or **RC** entries should be replicated as a caption in the appearance of the line, as shown in Figure 8.6 and Figure 8.7. The text should be rendered in a manner appropriate to the content, taking into account factors such as writing direction. 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* **Cap** *is true; PDF 1.7*) An array of two numbers specifying the offset of the caption text from its normal position. The first value is the horizontal offset along the annotation line from its midpoint, with a positive value indicating offset to the right and a negative value indicating offset to the left. The second value is the vertical offset perpendicular to the annotation line, with a positive value indicating a shift up and a negative value indicating a shift down. Default value: [0, 0] (no offset from normal positioning)

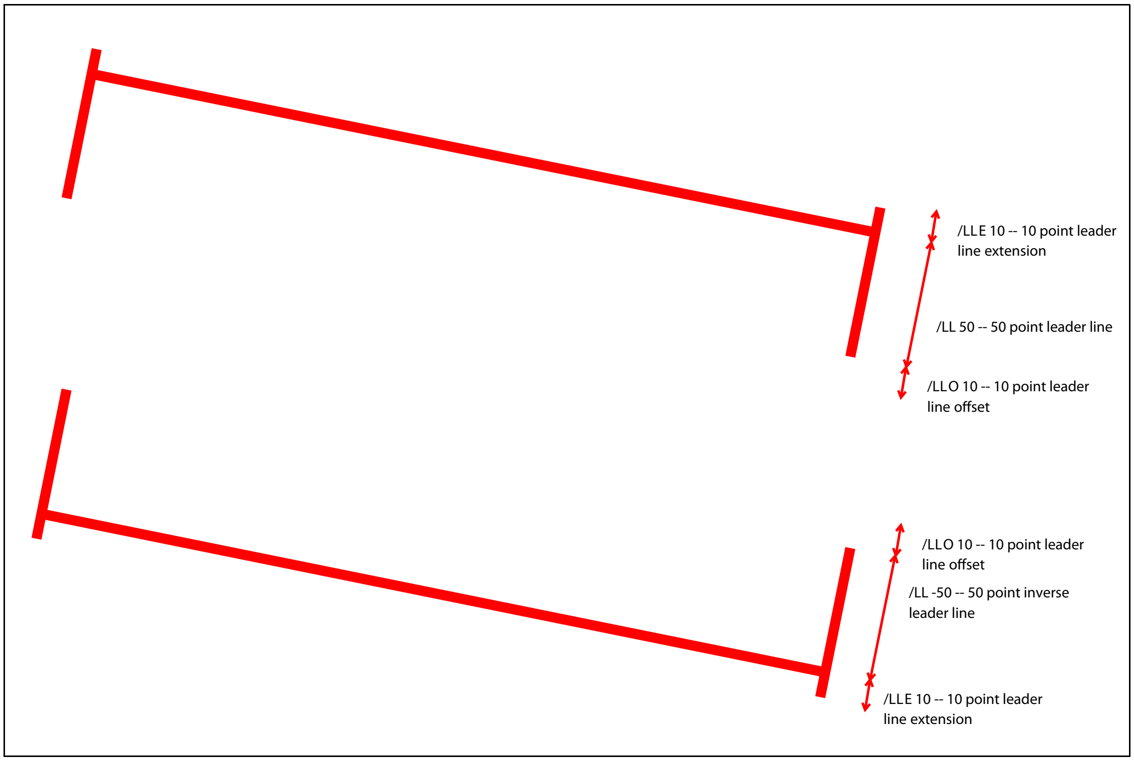

/LLE 10 -- 10 point leader line extension
/LL 50 -- 50 point leader line
/LLO 10 -- 10 point leader line offset
/LLO 10 -- 10 point leader line offset
/LL -50 -- 50 point inverse leader line
/LLE 10 -- 10 point leader line extension

FIGURE 8.5 *Leader lines*

Figure 8.6 illustrates the effect of including a caption to a line annotation, which is specified by setting **Cap** to true.

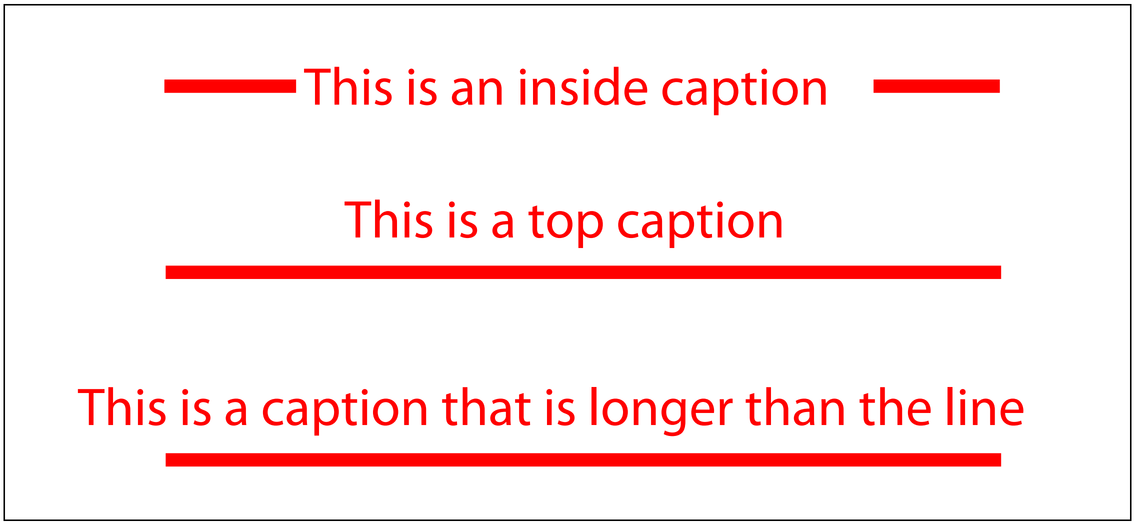

This is an inside caption
This is a top caption
This is a caption that is longer than the line

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.

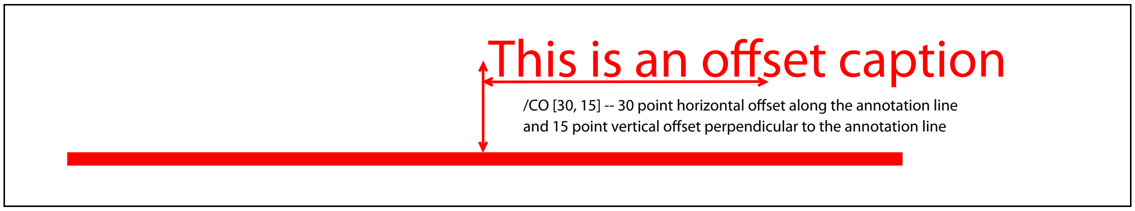
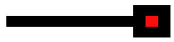
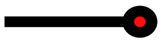
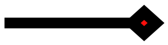
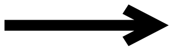
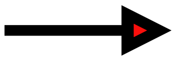
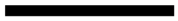
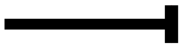
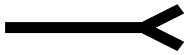
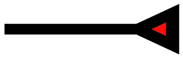
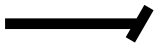

This is an offset caption
/CO [30, 15] -- 30 point horizontal offset along the annotation line
and 15 point vertical offset perpendicular to the annotation line

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

TABLE 8.27 Line ending styles

NAME	APPEARANCE	DESCRIPTION
Square		A square filled with the annotation's interior color, if any
Circle		A circle filled with the annotation's interior color, if any
Diamond		A diamond shape filled with the annotation's interior color, if any
OpenArrow		Two short lines meeting in an acute angle to form an open arrowhead
ClosedArrow		Two short lines meeting in an acute angle as in the OpenArrow style (see above) and connected by a third line to form a triangular closed arrowhead filled with the annotation's interior color, if any
None		No line ending
Butt		*(PDF 1.5)* A short line at the endpoint perpendicular to the line itself
ROpenArrow		*(PDF 1.5)* Two short lines in the reverse direction from OpenArrow
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 entries specific to these types of annotations.

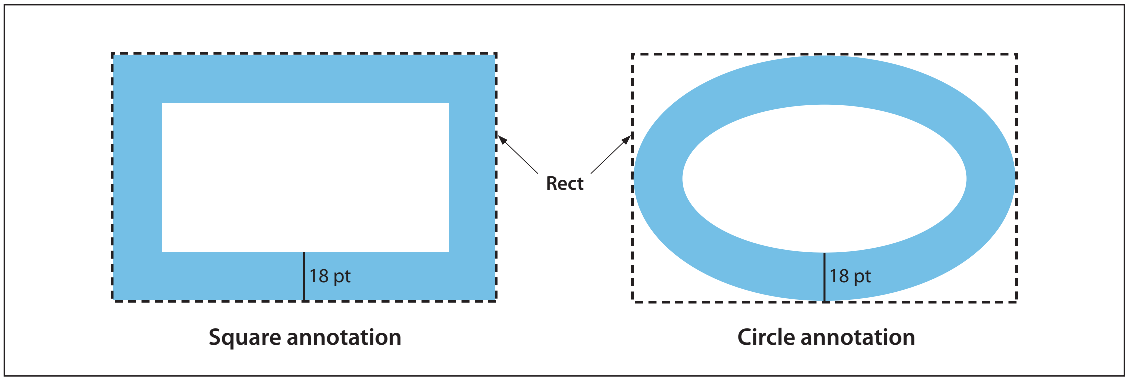


FIGURE 8.8 *Square and circle annotations*

TABLE 8.28 Additional entries specific to a square or circle annotation

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 border described by the **BS** entry (see Table 8.18).

KEY	TYPE	VALUE
RD	rectangle	*(Optional; PDF 1.5)* A set of four numbers describing the numerical differences between two rectangles: the **Rect** entry of the annotation and the actual boundaries of the underlying square or circle. Such a difference can occur in situations where a border effect (described by **BE**) causes the size of the **Rect** to increase beyond that of the square or circle. The four numbers correspond to the differences in default user space between the left, top, right, and bottom coordinates of **Rect** and those of the square or circle, respectively. Each value must be greater than or equal to 0. The sum of the top and bottom differences must be less than the height of **Rect**, and the sum of the left and right differences must be less than the width of **Rect**.

Polygon and Polyline Annotations

Polygon annotations (PDF 1.5) display closed polygons on the page. Such polygons may have any number of vertices connected by straight lines. *Polyline annotations (PDF 1.5)* are similar to polygons, except that the first and last vertex are 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 width and dash pattern to be used in drawing the line. ***Note:*** *The annotation dictionary's* ***AP*** *entry, if present, takes precedence over the* ***Vertices*** *and* ***BS*** *entries; see Table 8.15 on page 606 and Section 8.4.4, "Appearance Streams."*

KEY	TYPE	VALUE
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** 4 **DeviceCMYK**
BE	dictionary	*(Optional; meaningful only for polygon annotations)* A *border effect dictionary* describing an effect applied to the border described by the **BS** entry (see Table 8.18).
IT	name	*(Optional; PDF 1.6)* A name describing the intent of the polygon or polyline annotation (see also Table 8.21). The following values are valid: **PolygonCloud**, which means that the annotation is intended to function as a 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 text markup annotations

KEY	TYPE	VALUE
Subtype	name	*(Required)* The type of annotation that this dictionary describes; must be **Highlight**, **Underline**, **Squiggly**, or **StrikeOut** for a highlight, underline, squiggly-underline, or strikeout annotation, respectively.
QuadPoints	array	*(Required)* An array of $8 \times n$ numbers specifying the coordinates of n quadrilaterals in default user space. Each quadrilateral encompasses a word or group of contiguous words in the text underlying the annotation. The coordinates for each quadrilateral are given in the order $x_1\ y_1\ x_2\ y_2\ x_3\ y_3\ x_4\ y_4$ specifying the quadrilateral's four vertices in counterclockwise order (see Figure 8.9). The text is oriented with respect to the edge connecting points (x_1, y_1) and (x_2, y_2). (See implementation note 92 in Appendix H.) ***Note:*** *The annotation dictionary's* ***AP*** *entry, if present, takes precedence over* ***QuadPoints****; see Table 8.15 and Section 8.4.4, "Appearance Streams."*

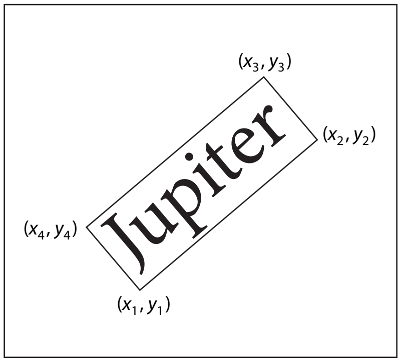


FIGURE 8.9 *QuadPoints specification*

Caret Annotations

A caret annotation *(PDF 1.5)* is a visual symbol that indicates the presence of text edits. Table 8.31 lists the entries specific to caret annotations.

TABLE 8.31 Additional entries specific to a caret annotation

KEY	TYPE	VALUE
Subtype	name	*(Required)* The type of annotation that this dictionary describes; must be **Caret** for a caret annotation.
RD	rectangle	*(Optional; PDF 1.5)* A set of four numbers describing the numerical differences between two rectangles: the **Rect** entry of the annotation and the actual boundaries of the underlying caret. Such a difference can occur, for example, when a paragraph symbol specified by **Sy** is displayed along with the caret. The four numbers correspond to the differences in default user space between the left, top, right, and bottom coordinates of **Rect** and those of the caret, respectively. Each value must be greater than or equal to 0. The sum of the top and bottom differences must be less than the height of **Rect**, and the sum of the left and right differences must be less than the width of **Rect**.
Sy	name	*(Optional)* A name specifying a symbol to be associated with the caret: P A new paragraph symbol (¶) should be associated with the caret. 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 to be used in displaying the annotation. Viewer applications should provide predefined icon appearances for at least the following standard names: Approved, AsIs, Confidential, Departmental, Draft, Experimental, Expired, Final, ForComment, ForPublicRelease, NotApproved, NotForPublicRelease, Sold, TopSecret. Additional names may be supported as well. Default value: Draft. ***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."*

Ink Annotations

An *ink annotation (PDF 1.3)* represents a freehand "scribble" composed of one or 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. ***Note:*** *The annotation dictionary's* ***AP*** *entry, if present, takes precedence over the* ***InkList*** *and* ***BS*** *entries; see Table 8.15 on page 606 and Section 8.4.4, "Appearance Streams."*

Pop-up Annotations

A *pop-up annotation (PDF 1.3)* displays text in a pop-up window for entry and editing. It typically does not appear alone but is associated with a markup annotation, its *parent annotation*, and is used for editing the parent's text. It has no appearance stream or associated actions of its own and is identified by the **Popup** entry in the parent's annotation dictionary (see Table 8.21 on page 618). Table 8.34 shows the annotation dictionary entries specific to this type of annotation.

TABLE 8.34 Additional entries specific to a pop-up annotation

KEY	TYPE	VALUE
Subtype	name	*(Required)* The type of annotation that this dictionary describes; must be **Popup** for a pop-up annotation.
Parent	dictionary	*(Optional; must be an indirect reference)* The parent annotation with which this pop-up annotation is associated. ***Note:*** *If this entry is present, the parent annotation's* ***Contents****,* ***M****,* ***C****, and* ***T*** *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 relating to the attached file. Viewer applications should use this entry rather than the optional **Desc** entry *(PDF 1.6)* in the file specification dictionary (see Table 3.41) identified by the annotation's **FS** entry; see implementation note 95 in Appendix H.

TABLE 8.35 Additional entries specific to a file attachment annotation

KEY	TYPE	VALUE
Subtype	name	*(Required)* The type of annotation that this dictionary describes; must be **FileAttachment** for a file attachment annotation.
FS	file specification	*(Required)* The file associated with this annotation.
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: Graph PushPin Paperclip Tag Additional names may be supported as well. Default value: PushPin. ***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."*

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.
Sound	stream	*(Required)* A sound object defining the sound to be played when the annotation is activated (see Section 9.2, "Sounds").

KEY	TYPE	VALUE
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 standard names Speaker and Mic. Additional names may be supported as well. Default value: Speaker. *__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."*

Movie Annotations

A *movie annotation (PDF 1.2)* contains animated graphics and sound to be presented on the computer screen and through the speakers. When the annotation is activated, the movie is played. Table 8.37 shows the annotation dictionary entries specific to this type of annotation. Movies are discussed in Section 9.3, "Movies."

TABLE 8.37 Additional entries specific to a movie annotation

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

A *screen annotation (PDF 1.5)* specifies a region of a page upon which media clips may be played. It also serves as an object from which actions can be triggered. "Rendition Actions" on page 668 discusses the relationship between screen annotations and rendition actions. Table 8.38 shows the annotation dictionary entries specific to this type of annotation.

TABLE 8.38 Additional entries specific to a screen annotation

KEY	TYPE	VALUE
Subtype	name	*(Required)* The type of annotation that this dictionary describes; must be **Screen** for a screen annotation.
T	text string	(*Optional*) The title of the screen annotation.
MK	dictionary	*(Optional)* An appearance characteristics dictionary (see Table 8.40). The **I** entry of this dictionary provides the icon used in generating the appearance referred to by the screen annotation's **AP** entry.
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 screen annotation's behavior in response to various trigger events (see Section 8.5.2, "Trigger 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. As a convenience, when a field has only a single associated widget annotation, the contents of the field dictionary (Section 8.6.2, "Field Dictionaries") and the annotation dictionary can be merged into a single dictionary containing entries that pertain to both a field and an annotation. (This presents no ambiguity, since the contents of the two kinds of dictionaries do not conflict.) Table 8.39 shows the annotation dictionary entries specific to this type of annotation; interactive forms and fields are discussed at length in Section 8.6.

TABLE 8.39 Additional entries specific to a widget annotation

KEY	TYPE	VALUE
Subtype	name	*(Required)* The type of annotation that this dictionary describes; must be **Widget** for a widget annotation.
H	name	*(Optional)* The annotation's *highlighting mode*, the visual effect to be used when the mouse button is pressed or held down inside its active area: N (None) No highlighting. I (Invert) Invert the contents of the annotation rectangle. O (Outline) Invert the annotation's border. P (Push) Display the annotation's down appearance, if any (see Section 8.4.4, "Appearance Streams"). If no down appearance is defined, offset the contents of the annotation rectangle to appear as if it were being 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").
BS	dictionary	*(Optional; PDF 1.2)* A border style dictionary (see Table 8.17 on page 611) specifying the width and dash pattern to be used in drawing the annotation's border. ***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."*

The **MK** entry can be used to provide an *appearance characteristics dictionary* containing additional information for constructing the annotation's appearance stream. Table 8.40 shows the contents of this dictionary.

TABLE 8.40 Entries in an appearance characteristics dictionary

KEY	TYPE	VALUE
R	integer	*(Optional)* The number of degrees by which the widget annotation is rotated counterclockwise relative to the page. The value must be a multiple of 90. Default value: 0.
BC	array	*(Optional)* An array of numbers in the range 0.0 to 1.0 specifying the color of the widget annotation's border. 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**
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	stream	*(Optional; pushbutton fields only; must be an indirect reference)* A form XObject defining the widget annotation's *normal icon*, displayed when it is not interacting with the user.

KEY	TYPE	VALUE
RI	stream	*(Optional; pushbutton fields only; must be an indirect reference)* A form XObject defining the widget annotation's *rollover icon*, displayed when the user rolls the cursor into its active area without pressing the mouse button.
IX	stream	*(Optional; pushbutton fields only; must be an indirect reference)* A form XObject defining the widget annotation's *alternate (down) icon*, displayed when the mouse button is pressed within its active area.
IF	dictionary	*(Optional; pushbutton fields only)* An icon fit dictionary (see Table 8.97 on page 719) specifying how to display the widget annotation's icon within its annotation rectangle. If present, the icon fit dictionary applies to all of the annotation's icons (normal, rollover, and alternate).
TP	integer	*(Optional; pushbutton fields only)* A code indicating where to position the text of the widget annotation's caption relative to its icon: 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

A *trap network annotation (PDF 1.3)* defines the trapping characteristics for a page of a PDF document. (*Trapping* is the process of adding marks to a page along color boundaries to avoid unwanted visual artifacts resulting from mis-registration of colorants when the page is printed.) A page may have at most one trap network annotation, whose **Subtype** entry has the value **TrapNet** and which is always the last element in the page object's **Annots** array (see "Page Objects" on page 144). See Section 10.10.5, "Trapping Support," for further discussion.

Watermark Annotations

A *watermark annotation (PDF 1.6)* is used to represent graphics that are expected to be printed at a fixed size and position on a page, regardless of the dimensions of the printed page. The **FixedPrint** entry of a watermark annotation dictionary (see Table 8.41) is a dictionary that contains values for specifying the size and position of the annotation (see Table 8.42).

Watermark annotations have no pop-up window or other interactive elements. When displaying a watermark annotation on-screen, viewer applications should use the dimensions of the media box as the page size so that the scroll and zoom behavior is the same as for other annotations.

Note: *Since many printing devices have nonprintable margins, it is recommended 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 print dictionary

KEY	TYPE	VALUE
Type	name	*(Required)* Must be **FixedPrint**.
Matrix	array	*(Optional)* The matrix used to transform the annotation's rectangle before rendering. Default value: the identity matrix [1 0 0 1 0 0]. ***Note:** When positioning content near the edge of a page, it is recommended that this entry be used to provide a reasonable offset to allow for nonprintable margins.*
H	number	*(Optional)* The amount to translate the associated content horizontally, as a percentage of the width of the target media (or if unknown, the width 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.
V	number	*(Optional)* The amount to translate the associated content vertically, as a percentage 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 user space, a new matrix is computed that cancels out B and translates the origin of the printed page to the origin of the default user space. This transformation is applied to ensure the correct scaling and alignment.

Example 8.10 shows a watermark annotation that prints a text string one inch from the left and one inch from the top of the printed page.

Example 8.10

```
8 0 obj                           % Watermark appearance
    <<
        /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
    >>
endobj
```

In situations other than the usual case where the PDF page size equals the printed page size, watermark annotations with a **FixedPrint** entry should be printed in the following manner:

- When page tiling is selected in a viewer application (that is, a single PDF page is printed on multiple pages), the annotations are printed at the specified size and position on each page to ensure that any enclosed content is present and legible on each printed page.
- When *n*-up printing is selected (that is, multiple PDF pages are printed on a single page), the annotations are printed at the specified size and are positioned as if the dimensions of the printed page were limited to a single portion of the page. This ensures that any enclosed content does not overlap content from other pages, thus rendering it illegible. (See implementation note 97 in Appendix 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 dictionaries. The dictionary may contain additional entries specific to a particular action type; see the descriptions of individual action types in Section 8.5.3, "Action Types," for details.

TABLE 8.43 Entries common to all action dictionaries

KEY	TYPE	VALUE
Type	name	*(Optional)* The type of PDF object that this dictionary describes; if present, must be **Action** for an action dictionary.
S	name	*(Required)* The type of action that this dictionary describes; see Table 8.48 on page 653 for specific values.
Next	dictionary or array	*(Optional; PDF 1.2)* The next action or sequence of actions to be performed after the action represented by this dictionary. The value is either a single action dictionary or an array of action dictionaries to be performed in order; see below for further discussion.

The action dictionary's **Next** entry *(PDF 1.2)* allows sequences of actions to be chained together. For example, the effect of clicking a link annotation with the mouse might be to play a sound, jump to a new page, and start up a movie. Note that the **Next** entry is not restricted to a single action but may contain an array of actions, each of which in turn may have a **Next** entry of its own. The actions may 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.*

8.5.2 Trigger Events

An annotation, page object, or (beginning with PDF 1.3) interactive form field may include an entry named **AA** that specifies an *additional-actions dictionary (PDF 1.2)* that extends the set of events that can trigger the execution of an action. In PDF 1.4, the document catalog dictionary (see Section 3.6.1, "Document Catalog") may also contain an **AA** entry for trigger events affecting the document as a whole. Tables 8.44 to 8.47 show the contents of this type of dictionary. (See implementation notes 98 and 99 in Appendix H.)

PDF 1.5 introduces four trigger events to support multimedia presentations:

- The **PO** and **PC** entries have a similar function to the **O** and **C** entries in the page object's additional-actions dictionary (see Table 8.45). However, associating these triggers with annotations allows annotation objects to be self-contained and greatly simplifies authoring. For example, annotations containing such actions can be copied or moved between pages without requiring page open/close actions to be changed.
- The **PV** and **PI** entries allow a distinction between pages that are open and pages that are visible. At any one time, only a single page is considered open in the viewer application, while more than one page may be visible, depending on the 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.") ***Note:** For backward compatibility, the **A** entry in an annotation dictionary, if present, takes precedence over this entry (see Table 8.15 on page 606).*
Fo	dictionary	*(Optional; PDF 1.2; widget annotations only)* An action to be performed when the annotation receives the input focus.

KEY	TYPE	VALUE
Bl	dictionary	*(Optional; PDF 1.2; widget annotations only)* (Uppercase B, lowercase L) An action to be performed when the annotation loses the input focus. (The name **Bl** stands for “blurred.”)
PO	dictionary	*(Optional; PDF 1.5)* An action to be performed when the page containing the annotation 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). The action is executed after the **O** action in the page’s additional-actions dictionary (see Table 8.45) and the **OpenAction** entry in the document catalog (see Table 3.25), if such actions are present.
PC	dictionary	*(Optional; PDF 1.5)* An action to be performed when the page containing the annotation is closed (for example, when the user navigates to the next or previous page, or follows a link annotation or outline item). The action is executed before the **C** action in the page’s additional-actions dictionary (see Table 8.45), if present.
PV	dictionary	*(Optional; PDF 1.5)* An action to be performed when the page containing the annotation becomes visible in the viewer application’s user interface.
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 additional-actions dictionary

KEY	TYPE	VALUE
K	dictionary	*(Optional; PDF 1.3)* A JavaScript action to be performed when the user types a keystroke into a text field or combo box or modifies the selection in a scrollable list box. This action can check the keystroke for validity and reject or modify it.
F	dictionary	*(Optional; PDF 1.3)* A JavaScript action to be performed before the field is formatted to display its current value. This action can modify the field's value before formatting.
V	dictionary	*(Optional; PDF 1.3)* A JavaScript action to be performed when the field's value is changed. This action can check the new value for validity. (The name **V** stands for "validate.")
C	dictionary	*(Optional; PDF 1.3)* A JavaScript action to be performed to recalculate the value of this field when that of another field changes. (The name **C** stands for "calculate.") The order in which the document's fields are recalculated is defined by the **CO** entry in the interactive form dictionary (see Section 8.6.1, "Interactive Form Dictionary").

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.")

For purposes of the trigger events **E** (enter), **X** (exit), **D** (down), and **U** (up), the term *mouse* denotes a generic pointing device with the following characteristics:

- A selection button that can be *pressed*, *held down*, and *released*. If there is more than one mouse button, the selection button is typically the left button.

- A notion of *location*—that is, an indication of where on the screen the device is pointing. Location is typically denoted by a screen cursor.
- A notion of *focus*—that is, which element in the document is currently interacting with the user. In many systems, this element is denoted by a blinking caret, a focus rectangle, or a color change.

PDF viewer applications must ensure the presence of such a device for the corresponding actions to be executed correctly. Mouse-related trigger events are subject to the following constraints:

- An **E** (enter) event can occur only when the mouse button is up.
- An **X** (exit) event cannot occur without a preceding **E** event.
- A **U** (up) event cannot occur without a preceding **E** and **D** event.
- In the case of overlapping or nested annotations, entering a second annotation's active area causes an **X** event to occur for the first annotation.

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

PDF supports the standard action types listed in Table 8.48. The following sections describe each of these types in detail. Plug-in extensions may add new action types.

TABLE 8.48 Action types

ACTION TYPE	DESCRIPTION	DISCUSSED IN SECTION
GoTo	Go to a destination in the current document.	"Go-To Actions" on page 654
GoToR	("Go-to remote") Go to a destination in another document.	"Remote Go-To Actions" on page 655
GoToE	("Go-to embedded"; *PDF 1.6*) Go to a destination in an embedded file.	"Embedded Go-To Actions" on page 655
Launch	Launch an application, usually to open a file.	"Launch Actions" on page 659
Thread	Begin reading an article thread.	"Thread Actions" on page 661
URI	Resolve a uniform resource identifier.	"URI Actions" on page 662
Sound	*(PDF 1.2)* Play a sound.	"Sound Actions" on page 663
Movie	*(PDF 1.2)* Play a movie.	"Movie Actions" on page 664
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 670

Note: *Previous versions of the PDF specification described an action type known as the* set-state action*; this type of action is now considered obsolete and its use is no longer recommended. An additional action type, the* no-op action*, was defined in PDF 1.2 but never implemented; it is no longer defined and should be ignored.*

Go-To Actions

A *go-to action* changes the view to a specified destination (page, location, and magnification factor). Table 8.49 shows the action dictionary entries specific to this type of action.

TABLE 8.49 Additional entries specific to a go-to action

KEY	TYPE	VALUE
S	name	*(Required)* The type of action that this dictionary describes; must be **GoTo** for a go-to action.
D	name, byte string, or array	*(Required)* The destination to jump to (see Section 8.2.1, "Destinations").

Specifying a go-to action in the **A** entry of a link annotation or outline item (see Tables 8.24 on page 622 and 8.4 on page 585) has the same effect as specifying the 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]
           >>
   >>
endobj
```

Remote Go-To Actions

A *remote go-to action* is similar to an ordinary go-to action but jumps to a destination in another PDF file instead of the current file. Table 8.50 shows the action dictionary entries specific to this type of action.

***Note:** Remote go-to actions cannot be used with embedded files; see "Embedded Go-To Actions" on page 655".*

TABLE 8.50 Additional entries specific to a remote go-to action

KEY	TYPE	VALUE
S	name	*(Required)* The type of action that this dictionary describes; must be **GoToR** for a remote go-to action.
F	file specification	*(Required)* The file in which the destination is located.
D	name, byte string, or array	*(Required)* The destination to jump to (see Section 8.2.1, "Destinations"). If the value is an array defining an explicit destination (as described under "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 dictionary entries specific to embedded go-to actions.

Embedded go-to actions provide a complete facility for linking between a file in a hierarchy of nested embedded files and another file in the same or different hierarchy. The following terminology is used:

- The *source* is the document containing the embedded go-to action.
- The *target* is the document in which the destination lives.

  The **T** entry in the action dictionary is a target dictionary that locates the target in relation to the source, in much the same way that a relative path describes the physical relationship between two files in a file system. Target dictionaries may be nested recursively to specify one or more intermediate targets before reaching the final one. As the hierarchy is navigated, each intermediate target is referred to as the *current document*. Initially, the source is the current document.

  Note: *It is an error for a target dictionary to have an infinite cycle (for example, one where a target dictionary refers to itself). Viewer applications should attempt to detect such cases and refuse to execute the action if found.*

- 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, or array	*(Required)* The destination in the target to jump to (see Section 8.2.1, “Destinations”).
NewWindow	boolean	*(Optional)* If **true**, the destination document should be opened in a new window; if **false**, the destination document should replace the current document in the same window. If this entry is absent, the viewer application should honor the current user preference.

KEY	TYPE	VALUE
T	dictionary	*(Optional if **F** is present; otherwise required)* A *target dictionary* (see Table 8.52) specifying path information to the target document. Each target dictionary specifies one element in the full path to the target and may have nested target dictionaries specifying additional elements.

TABLE 8.52 Entries specific to a target dictionary

KEY	TYPE	VALUE
R	name	*(Required)* Specifies the relationship between the current document and the target (which may be an intermediate target). Valid values are **P** (the target is the parent of the current document) and **C** (the target is a child of the current document).
N	byte string	*(Required if the value of **R** is **C** and the target is located in the **EmbeddedFiles** name tree; otherwise, it must be absent)* The name of the file in the **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 relationships between source and target. Each object shown is an action dictionary for an embedded go-to action.

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)
      /F (someFile.pdf)
      /T <<   /R /C
            /N (Embedded document) >>
   >>
endobj
```

```
5 0 obj                          % Link from an embedded file to a normal file
    << /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
```

Launch Actions

A *launch action* launches an application or opens or prints a document. Table 8.53 shows the action dictionary entries specific to this type of action.

The optional **Win**, **Mac**, and **Unix** entries allow the action dictionary to include platform-specific parameters for launching the designated application. If no such entry is present for the given platform, the **F** entry is used instead. Table 8.54 shows the platform-specific launch parameters for the Windows platform. Parameters for the Mac OS and UNIX platforms are not yet defined at the time of publication.

TABLE 8.53 Additional entries specific to a launch action

KEY	TYPE	VALUE
S	name	*(Required)* The type of action that this dictionary describes; must be **Launch** for a launch action.
F	file specification	*(Required if none of the entries **Win**, **Mac**, or **Unix** is present)* The application to be launched or the document to be opened or printed. If this entry is absent and the viewer application does not understand any of the alternative entries, it should do nothing.
Win	dictionary	*(Optional)* A dictionary containing Windows-specific launch parameters (see 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 to be opened or printed, in standard Windows pathname format. If the name string includes a backslash character (\\), the backslash must itself be preceded by a backslash. ***Note:** This value must be a simple string; it is not a file specification.*

KEY	TYPE	VALUE
D	byte string	*(Optional)* A bye string specifying the default directory in standard DOS syntax.
O	ASCII string	*(Optional)* An ASCII string specifying the operation to perform: open Open a document. print Print a document. If the **F** entry designates an application instead of a document, this entry is ignored and the application is launched. Default value: open.
P	byte string	*(Optional)* A parameter string to be passed to the application designated by the **F** entry. This entry should be omitted if **F** designates a document.

Thread Actions

A *thread action* jumps to a specified bead on an article thread (see Section 8.3.2, "Articles"), in either the current document or a different one. Table 8.55 shows 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. • The title of the thread as specified in its thread information dictionary (see Table 8.11 on page 596). If two or more threads have the same title, the one appearing first in the document catalog's **Threads** array is used.

KEY	TYPE	VALUE
B	dictionary or integer	*(Optional)* The bead in the destination thread, specified in one of the following forms: • An indirect reference to a bead dictionary (see Section 8.3.2, "Articles"). In this case, the thread must be in the current file. • The index of the bead within its thread. The first bead in a thread has index 0.

URI Actions

A *uniform resource identifier* (URI) is a string that identifies (*resolves* to) a resource on the Internet—typically a file that is the destination of a hypertext link, although it can also resolve to a query or other entity. (URIs are described in Internet RFC 2396, *Uniform Resource Identifiers (URI): Generic Syntax*; see the Bibliography.)

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.

If the **IsMap** flag is **true** and the user has triggered the URI action by clicking an annotation, the coordinates of the mouse position at the time the action is performed should be transformed from device space to user space and then offset relative to the upper-left corner of the annotation rectangle (that is, the value of the **Rect** entry in the annotation with which the URI action is associated). For example, if the mouse coordinates in user space are (x_m, y_m) and the annotation rectangle extends from (ll_x, ll_y) at the lower-left to (ur_x, ur_y) at the upper-right, 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 fractional, they should be rounded to the nearest integer values. They are then appended to the URI to be resolved, separated by commas and preceded by a question mark, as shown in this example:

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

To support URI actions, a PDF document's catalog (see Section 3.6.1, "Document Catalog") may include a **URI** entry whose value is a *URI dictionary.* At the time of publication, only one entry is defined for such a dictionary (see Table 8.57).

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.

Sound Actions

A *sound action (PDF 1.2)* plays a sound through the computer's speakers. Table 8.58 shows the action dictionary entries specific to this type of action. Sounds are discussed in Section 9.2, "Sounds."

TABLE 8.58 Additional entries specific to a sound action

KEY	TYPE	VALUE
S	name	*(Required)* The type of action that this dictionary describes; must be **Sound** for a sound action.
Sound	stream	*(Required)* A sound object defining the sound to be played (see Section 9.2, "Sounds"; see also implementation note 104 in Appendix H).
Volume	number	*(Optional)* The volume at which to play the sound, in the range −1.0 to 1.0; see implementation note 106 in Appendix H. Default value: 1.0.
Synchronous	boolean	*(Optional)* A flag specifying whether to play the sound synchronously or asynchronously; see implementation note 106 in Appendix H. If this flag is **true**, the viewer application retains control, allowing no further user interaction other than canceling the sound, until the sound has been completely played. Default value: **false**.
Repeat	boolean	*(Optional)* A flag specifying whether to repeat the sound indefinitely. If this entry is present, the **Synchronous** entry is ignored. Default value: **false**.
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 be displayed before attempting to play the movie; such page change actions must be done explicitly.*

The contents of a movie action dictionary are identical to those of a movie activation dictionary (see Table 9.31 on page 785), with the additional entries shown in Table 8.59. The contents of the activation dictionary associated with the movie annotation provide the default values. Any information specified in the movie action dictionary overrides these values.

TABLE 8.59 Additional entries specific to a movie action

KEY	TYPE	VALUE
S	name	*(Required)* The type of action that this dictionary describes; must be **Movie** for a movie action.
Annotation	dictionary	*(Optional)* An indirect reference to a movie annotation identifying the movie to be played.
T	text string	*(Optional)* The title of a movie annotation identifying the movie to be played. ***Note:** The dictionary must include either an **Annotation** or a **T** entry but not both.*
Operation	name	*(Optional)* The operation to be performed on the movie: Play — Start playing the movie, using the play mode specified by the dictionary's **Mode** entry (see Table 9.31 on page 785). If the movie is currently paused, it is repositioned to the beginning before playing (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 show and hide the annotation when the user rolls the cursor in and out of its active area on the page. This can be used to pop up a help label, or tool tip, describing the effect of clicking at that location on the page. Table 8.60 shows the action dictionary entries specific to this type of action. (See implementation notes 109 and 110 in Appendix H.)

TABLE 8.60 Additional entries specific to a hide action

KEY	TYPE	VALUE
S	name	*(Required)* The type of action that this dictionary describes; must be **Hide** for a hide action.
T	dictionary, text string, or array	*(Required)* The annotation or annotations to be hidden or shown, specified in any of the following forms: • An indirect reference to an annotation dictionary • A text string giving the fully qualified field name of an interactive form field whose associated widget annotation or annotations are to be affected (see "Field Names" on page 676) • An array of such dictionaries or text strings
H	boolean	*(Optional)* A flag indicating whether to hide the annotation (**true**) or show it (**false**). 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 any document using them is not portable. If the viewer encounters a named action that is inappropriate for a viewing platform, or if the viewer does not recognize the name, it should take no action.*

Table 8.62 shows the action dictionary entries specific to named actions.

TABLE 8.62 Additional entries specific to named actions

KEY	TYPE	VALUE
S	name	*(Required)* The type of action that this dictionary describes; must be **Named** for a named action.
N	name	*(Required)* The name of the action to be performed (see Table 8.61).

Set-OCG-State Actions

A *set-OCG-state action (PDF 1.5)* sets the state of one or more optional content groups (see Section 4.10, "Optional Content"). Table 8.63 shows the action dictionary entries specific to this type of action.

TABLE 8.63 Additional entries specific to a set-OCG-state action

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. If **PreserveRB** is **false**, radio-button state relationships, if any, are ignored. Default value: **true**.

When a set-OCG-state action is performed, the **State** array is processed from left to right. Each name is applied to subsequent groups in the array until the next name is encountered, as shown in the following example.

Example 8.13

```
<< /S /SetOCGState
   /State [/OFF 2 0 R 3 0 R /Toggle 16 0 R 19 0 R /ON 5 0 R]
>>
```

A group can appear more than once in the **State** array; its state is set each time it is encountered, based on the most recent name. For example, if the array contained [/OFF 1 0 R /Toggle 1 0 R], the group's state would be **ON** after the action was performed. **ON**, **OFF** and **Toggle** sequences have no required order. More than one sequence in the array may contain the same name.

Note: *While the specification allows a group to appear more than once in the* ***State*** *array, this is not intended to implement animation or any other sequential drawing operations. PDF processing applications are free to accumulate all state changes and 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 lists the entries in a rendition action dictionary.

TABLE 8.64 Additional entries specific to a rendition action

KEY	TYPE	VALUE
S	name	*(Required)* The type of action that this dictionary describes; must be **Rendition** for a rendition action.
R	dictionary	(*Required when **OP** is present with a value of 0 or 4; otherwise optional*) A rendition object (see Section 9.1.2, "Renditions").
AN	dictionary	(*Required if **OP** is present with a value of 0, 1, 2, 3 or 4; otherwise optional*) An indirect reference to a screen annotation (see "Screen Annotations" on page 639).
OP	integer	(*Required if **JS** is not present; otherwise optional*) The operation to perform when the action is triggered. Valid values are: 0 If no rendition is associated with the annotation specified by **AN**, play the rendition specified by **R**, associating it with the annotation. If a rendition is already associated with the annotation, it is stopped, and the new rendition is associated with the annotation. 1 Stop any rendition being played in association with the annotation specified 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 JavaScripts. If **OP** has an unrecognized value and there is no **JS** entry, the action is invalid.

In some situations, a pause (**OP** value of 2) or resume (**OP** value of 3) operation may not make sense (for example, for a JPEG image) or the player may not support it. In such cases, the user should be notified of the failure to perform the operation.

Before a rendition action is executed, the viewer application must make sure that the **P** entry of the screen annotation dictionary references a valid page object and that the annotation is present in the page object's **Annots** array (see Table 3.27).

A rendition may play in the rectangle occupied by a screen annotation, even if the annotation itself is not visible; for example, if its Hidden or NoView flags (see Table 8.16) are set. If a screen annotation is not visible because its location on the page is not being displayed by the viewer, the rendition is not visible. However, it may become visible if the view changes, such as by scrolling.

Transition Actions

A *transition action (PDF 1.5)* can be used to control drawing during a sequence of actions. As discussed in Section 8.5.1, "Action Dictionaries," the **Next** entry in an action dictionary can specify a sequence of actions. Viewer applications should normally suspend drawing when such a sequence begins and resume drawing when it ends. If a transition action is present during a sequence, the viewer should render the state of the page viewing area as it exists after completion of the previous 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.

TABLE 8.66 Additional entries specific to a go-to-3D-view action

KEY	TYPE	VALUE
S	name	*(Required)* The type of action that this dictionary describes; must be **GoTo3DView** for a transition action.

KEY	TYPE	VALUE
TA	dictionary	*(Required)* The target annotation for which to set the view.
V	(various)	*(Required)* The view to use. It can be one of the following types: • A 3D view dictionary (see Section 9.5.3, "3D Views"). • An integer specifying an index into the **VA** array in the 3D stream (see Table 9.35). • A text string matching the **IN** entry in one of the views in the **VA** array (see Table 9.39). • A name that indicates the first (**F**), last (**L**), next (**N**), previous (**P**), or default (**D**) entries in the **VA** array; see discussion below.

The **V** entry selects the view to apply to the annotation specified by **TA**. This view may be one of the predefined views specified by the **VA** entry of the 3D stream (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 document; see Section 8.6.4, "Form Actions."

Note: *Interactive forms should not be confused with form XObjects (see Section 4.9, "Form XObjects"). Despite the similarity of names, the two are different, unrelated types of objects.*

Each field in a document's interactive form is defined by a *field dictionary* (see Section 8.6.2, "Field Dictionaries"). For purposes of definition and naming, the fields can be organized hierarchically and can inherit attributes from their ancestors in the field hierarchy. A field's children in the hierarchy may also include widget annotations (see "Widget Annotations" on page 640) that define its appearance on the page. A field whose children are widget annotations is called a *terminal field.*

As a convenience, when a field has only a single associated widget annotation, the contents of the field dictionary and the annotation dictionary (Section 8.4.1, "Annotation Dictionaries") may be merged into a single dictionary containing entries that pertain to both a field and an annotation. (This presents no ambiguity, since the contents of the two kinds of dictionaries do not conflict.) If such an object defines an appearance stream, the appearance must be consistent with the object's current value as a field.

Note: *Fields containing text whose contents are not known in advance may need to construct their appearance streams dynamically instead of defining them statically 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 document (see "Variable Text" on page 677). Default value: **false**.
SigFlags	integer	*(Optional; PDF 1.3)* A set of flags specifying various document-level characteristics related to signature fields (see Table 8.68, below, and "Signature Fields" on page 695). Default value: 0.

KEY	TYPE	VALUE
CO	array	*(Required if any fields in the document have additional-actions dictionaries containing a* **C** *entry; PDF 1.3)* An array of indirect references to field dictionaries with calculation actions, defining the *calculation order* in which their values will be recalculated when the value of any field changes (see Section 8.5.2, "Trigger Events").
DR	dictionary	*(Optional)* A resource dictionary (see Section 3.7.2, "Resource Dictionaries") containing default resources (such as fonts, patterns, or color spaces) to be used by form field appearance streams. At a minimum, this dictionary must contain a **Font** entry specifying the resource name and font dictionary of the default font for displaying text. (See implementation notes 113 and 114 in Appendix H.)
DA	string	*(Optional)* A document-wide default value for the **DA** attribute of variable text fields (see "Variable Text" on page 677).
Q	integer	*(Optional)* A document-wide default value for the **Q** attribute of variable text fields (see "Variable Text" on page 677).
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 the meanings of the flags; all undefined flag bits are reserved and must be set to 0.

TABLE 8.68 Signature flags

BIT POSITION	NAME	MEANING
1	SignaturesExist	If set, the document contains at least one signature field. This flag allows a viewer application to enable user interface items (such as menu items or pushbuttons) related to signature processing without having to scan the entire document for the presence of signature fields.
2	AppendOnly	If set, the document contains signatures that may be invalidated if the file is saved (written) in a way that alters its previous contents, as opposed to an incremental update. Merely updating the file by appending new information to the end of the previous version is safe (see Section G.6, "Updating Example"). Viewer applications can use this flag to present a user requesting a full save with an additional alert box warning that signatures will be invalidated and requiring explicit confirmation before continuing 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 field dictionaries

KEY	TYPE	VALUE
FT	name	*(Required for terminal fields; inheritable)* The type of field that this dictionary describes: **Btn** Button (see "Button Fields" on page 685) **Tx** Text (see "Text Fields" on page 691) **Ch** Choice (see "Choice Fields" on page 693) **Sig** *(PDF 1.3)* Signature (see "Signature Fields" on page 695) ***Note:*** *This entry may be present in a nonterminal field (one whose descendants are fields) to provide an inheritable* **FT** *value. However, a nonterminal field does not logically have a type of its own; it is merely a container for inheritable attributes that are intended for descendant terminal fields of any type.*
Parent	dictionary	*(Required if this field is the child of another in the field hierarchy; absent otherwise)* The field that is the immediate parent of this one (the field, if any, whose **Kids** array includes this field). A field can have at most one parent; that is, it can be included in the **Kids** array of at most one other field.
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").
TM	text string	*(Optional; PDF 1.3)* The *mapping name* to be used when exporting interactive form field data from the document.
Ff	integer	*(Optional; inheritable)* A set of flags specifying various characteristics of the field (see Table 8.70). Default value: 0.

KEY	TYPE	VALUE
V	(various)	*(Optional; inheritable)* The field's value, whose format varies depending on the field type. See the descriptions of individual field types for further information.
DV	(various)	*(Optional; inheritable)* The default value to which the field reverts when a reset-form action is executed (see "Reset-Form Actions" on page 707). The format of this value is the same as that of **V**.
AA	dictionary	*(Optional; PDF 1.2)* An additional-actions dictionary defining the field's behavior in response to various trigger events (see Section 8.5.2, "Trigger Events"). This entry has exactly the same meaning as the **AA** entry in an annotation dictionary (see Section 8.4.1, "Annotation Dictionaries").

The value of the field dictionary's **Ff** entry is an unsigned 32-bit integer containing flags specifying various characteristics of the field. Bit positions within the flag word are numbered from 1 (low-order) to 32 (high-order). The flags shown in Table 8.70 are common to all types of fields. Flags that apply only to specific field types are discussed in the sections describing those types. All undefined flag 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).

Field Names

The **T** entry in the field dictionary (see Table 8.69 on page 675) holds a text string defining the field's *partial field name*. The *fully qualified field name* is not explicitly defined but is constructed from the partial field names of the field and all of its ancestors. For a field with no parent, the partial and fully qualified names are the same. For a field that is the child of another field, the fully qualified name is formed by appending the child field's partial name to the parent's fully qualified name, separated by a period (.):

parent's_full_name.child's_partial_name

For example, if a field with the partial field name PersonalData has a child whose partial name is Address, which in turn has a child with the partial name ZipCode, the fully qualified name of this last field is

PersonalData.Address.ZipCode

Thus, all fields descended from a common ancestor share the ancestor's fully qualified field name as a common prefix in their own fully qualified names.

It is possible for different field dictionaries to have the same fully qualified field name if they are descendants of a common ancestor with that name and have no 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.

In such cases, the PDF document cannot provide a statically defined appearance stream for displaying the field. Instead, the viewer application must construct an appearance stream dynamically at viewing time. The dictionary entries shown in Table 8.71 provide general information about the field's appearance that can be combined with the specific text it contains to construct an appearance stream.

TABLE 8.71 Additional entries common to all fields containing variable text

KEY	TYPE	VALUE
DA	string	*(Required; inheritable)* The *default appearance string* containing a sequence of valid page-content graphics or text state operators that define such properties as the field's text size and color.
Q	integer	*(Optional; inheritable)* A code specifying the form of *quadding* (justification) to be 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 string, as described in "Rich Text Strings" on page 680.
RV	text string or text stream	*(Optional; PDF 1.5)* A rich text string, as described in "Rich Text Strings" on page 680.

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 interactive form dictionary's **DR** entry (see Table 8.67); see also implementation note 118 in Appendix H.
- The lower-left corner of the bounding box (**BBox**) is set to coordinates (0, 0) in the form coordinate system. The box's top and right coordinates are taken from

the dimensions of the annotation rectangle (the **Rect** entry in the widget annotation dictionary).

- All other entries in the appearance stream's form dictionary are set to their default values (see Section 4.9, "Form XObjects").

The appearance stream includes the following section of marked content, which represents the portion of the stream that draws the text:

Example 8.14

```
/Tx BMC                                                      % Begin marked content with tag Tx
    q                                                        % Save graphics state
        …Any required graphics state changes, such as clipping…
        BT                                                   % Begin text object
            …Default appearance string (DA)…
            …Text-positioning and text-showing operators to show the variable text…
        ET                                                   % End text object
    Q                                                        % Restore graphics state
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.

The default appearance string should contain at most one **Tm** (text matrix) operator. If this operator is present, the viewer application should replace the horizontal and vertical translation components with positioning values it determines to be appropriate, based on the field value, the quadding (**Q**) attribute, and any layout rules it employs. If the default appearance string contains no **Tm** operator, the viewer should insert one in the appearance stream (with appropriate horizontal and vertical translation components) after the default appearance string and before the text-positioning and text-showing operators for the variable text.

To update an existing appearance stream to reflect a new field value, the viewer application should first copy any needed resources from the document's **DR** dictionary (see Table 8.67) into the stream's **Resources** dictionary. (If the **DR** and **Resources** dictionaries contain resources with the same name, the one already in the **Resources** dictionary should be left intact, *not* replaced with the corresponding value from the **DR** dictionary.) The viewer application should then replace the existing contents of the appearance stream from /Tx BMC to the matching EMC with the corresponding new contents as shown in Example 8.14. (If the existing appearance stream contains no marked content with tag Tx, the new contents should be appended to the end of the original stream.) Also see implementation 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 <span>) 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 text elements and attributes are a superset of those described in Table 8.72, Table 8.73 and Table 8.73. In PDF 1.7, PDF supports the rich text elements and attributes specified in the *XML Forms Architecture (XFA) Specification, 2.4* (see Bibliography). XFA 2.2 and XFA 2.4 describe the same rich text elements and attributes; however, XFA 2.4 expands the range of supported character codes.

TABLE 8.72 XHTML elements used in rich text strings

ELEMENT	DESCRIPTION
<body>	The element at the root of the XML document. Table 8.73 lists the required attributes for this element.
<p>	Encloses text that is interpreted as a paragraph. It may take the style attributes listed in Table 8.74.
<i>	Encloses text that is displayed in an italic font.
<b>	Encloses text that is displayed in a bold font.
<span>	Groups text solely for the purpose of applying styles (using the attributes in Table 8.74).

TABLE 8.73 Attributes of the <body> element

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 string complies. PDF 1.5 supports XFA 2.0; PDF 1.6 supports XFA 2.2; and PDF 1.7 supports XFA 2.4.

TABLE 8.74 CSS2 style attributes used in rich text strings

ATTRIBUTE	VALUE	DESCRIPTION
text-align	keyword	Horizontal alignment. Possible values: left, right, and center.
vertical-align	decimal	An amount by which to adjust the baseline of the enclosed text. A positive value indicates a superscript; a negative value indicates a subscript. The value is of the form <*decimal number*>pt, optionally preceded by a sign, and followed by "pt". Examples: -3pt, 4pt.
font-size	decimal	The font size of the enclosed text. The value is of the form <*decimal number*>pt.
font-style	keyword	Specifies whether the enclosed text should be displayed using a normal or italic (oblique) font. Possible values: normal, italic.
font-weight	keyword	The weight of the font for the enclosed text. Possible values: normal, bold, 100, 200, 300, 400, 500, 600, 700, 800, 900. ***Note:*** *normal is equivalent to 400, and bold is equivalent to 700.*
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.
font-stretch	keyword	Specifies a normal, condensed or extended face from a font family. Supported values from narrowest to widest are ultra-condensed, extra-condensed, condensed, semi-condensed, normal, semi-expanded, expanded, extra-expanded, and ultra-expanded.

Rich text strings are specified by the **RV** entry of variable text form field dictionaries (see Table 8.71) and the **RC** entry of markup annotation dictionaries (see Table 8.21). Rich text strings may be packaged as *text streams* (see Section 3.8.2, "Text Streams"). Form fields using rich text streams should also have the RichText flag set (see Table 8.77).

A *default style string* is specified by the **DS** entry for free text annotations (see Table 8.25) or variable text form fields (see Table 8.71). This string specifies the default values for style attributes, which are used for any style attributes that are not explicitly specified for the annotation or field. All attributes listed in Table 8.74 are legal in the default style string. This string, in addition to the **RV** or **RC** entry, is used to generate the appearance. The following entries are ignored by PDF 1.5-compliant viewers: the **Contents** entry for annotations, the **DA** entry for free text annotations, and the **V**, **DA**, and **Q** entries for form fields.

Note: *Markup annotations other than free text annotations (see "Markup Annotations" on page 616) do not use a default style string because their appearances are implemented using platform controls requiring the viewer application to pick an appropriate 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 &#13;) should be converted to their character equivalents.

   Note: *No attempt should be made to preserve formatting specified with markup elements. For example, although the <p> element implies a new line, a carriage return should not be generated in the associated flat text.*

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

If the strings are unequal, it is assumed the field has been modified by an older viewer, and a new rich text string should be created from the flat text.

When a rich text string specifies font attributes, the viewer application should use font name selection as described in section 15.3 of the CSS2 specification (see the Bibliography). It is strongly recommended that precedence be given to the fonts in the default resources dictionary, as specified by the **DR** entry in Table 8.67; see Implementation note 120 in Appendix H.

The following example illustrates the entries in a widget annotation dictionary for rich text. The **DS** entry specifies the default font. The **RV** entry contains two paragraphs of rich text: the first paragraph specifies bold and italic text in the default font; the second paragraph changes the font size.

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 types:

- *Button fields* represent interactive controls on the screen that the user can manipulate with the mouse. They include *pushbuttons*, *check boxes*, and *radio buttons*.
- *Text fields* are boxes or spaces in which the user can enter text from the keyboard.
- *Choice fields* contain several text items, at most one of which may be selected as the field value. They include scrollable *list boxes* and *combo boxes*.
- *Signature fields* represent electronic signatures for authenticating the identity of a user and the validity of the document's contents.

The following sections describe each of these field types in detail. Further types 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 shown in Table 8.75.

TABLE 8.75 Field flags specific to button fields

BIT POSITION	NAME	MEANING
15	NoToggleToOff	*(Radio buttons only)* If set, exactly one radio button must be selected at all times; clicking the currently selected button has no effect. If clear, clicking the selected button deselects it, leaving no button selected.
16	Radio	If set, the field is a set of radio buttons; if clear, the field is a check box. This flag is meaningful only if the Pushbutton flag is clear.
17	Pushbutton	If set, the field is a pushbutton that does not retain a permanent value.
26	RadiosInUnison	*(PDF 1.5)* If set, a group of radio buttons within a radio button field that use the same value for the on state will turn on and off in unison; that is if one is checked, they are all checked. If clear, the buttons are mutually exclusive (the same behavior as HTML radio buttons).

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.

The **V** entry in the field dictionary (see Table 8.69 on page 675) holds a name object representing the check box's appearance state, which is used to select the appropriate appearance from the appearance dictionary.

Example 8.16 shows a typical check box definition.

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
        ET
    Q
endstream
endobj
```

Beginning with PDF 1.4, the field dictionary for check boxes and radio buttons contains an optional **Opt** entry (see Table 8.76), which holds an array of text strings representing the export value of each annotation in the field. It is used for the following purposes:

- To represent the export values of check box and radio button fields in non-Latin writing systems. Because name objects in the appearance dictionary are limited to **PDFDocEncoding**, they cannot represent non-Latin text.
- To allow radio buttons or check boxes to be checked independently, even if they have the same export value.

  An example is a group of check boxes that are duplicated on more than one page, and the desired behavior is that when a user checks a box, the corresponding boxes on each of the other pages is also checked. In this case, each of the corresponding check boxes is a widget in the **Kids** array of a check box field.

  Note: *For radio buttons, the same behavior occurs only if the RadiosInUnison flag is set. If it is not set, at most one radio button in a field can be set at a time. See 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

A *radio button field* is a set of related buttons. Like check boxes, individual radio buttons have two states, on and off. A single radio button may not be turned off directly but only as a result of another button being turned on. Typically, a set of radio buttons (annotations that are children of a single radio button field) have at most one button in the on state at any given time; selecting any of the buttons automatically deselects all the others.

Note: *An exception occurs when multiple radio buttons in a field have the same on state and the RadiosInUnison flag is set. In that case, turning on one of the buttons turns on all of them.*

The field type is **Btn**, the Pushbutton flag (see Table 8.75 on page 686) is clear, and the Radio flag is set. This type of button field has an additional flag, NoToggle-ToOff, which specifies, if set, that exactly one of the radio buttons must be selected at all times. In this case, clicking the currently selected button has no effect; if the NoToggleToOff flag is clear, clicking the selected button deselects it, leaving no button selected.

The **Kids** entry in the radio button field's field dictionary (see Table 8.69 on page 675) holds an array of widget annotations representing the individual buttons in the set. The parent field's **V** entry holds a name object corresponding to the appearance state of whichever child field is currently in the on state; the default value 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
      /AP  <<  /N  <<  /MasterCard  8 0 R
                       /Off  9 0 R
                   >>
           >>
   >>
endobj
```

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

8 0 obj                                         % Appearance stream for "on" state
   << /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
   Q
endstream
endobj
```

Like a check box field, a radio button field can use the optional **Opt** entry in the field dictionary *(PDF 1.4)* to define export values for its constituent radio buttons, using Unicode encoding for non-Latin characters (see Table 8.76). **Opt** holds an array of text strings corresponding to the widget annotations representing the individual buttons in the field's **Kids** array.

Text Fields

A *text field* (field type **Tx**) is a box or space in which the user can enter text from the keyboard. The text may be restricted to a single line or may be permitted to span multiple lines, depending on the setting of the Multiline flag in the field dictionary's **Ff** entry. Table 8.77 shows the flags pertaining to this type of field.

TABLE 8.77 Field flags specific to text fields

BIT POSITION	NAME	MEANING
13	Multiline	If set, the field can contain multiple lines of text; if clear, the field's text is restricted to a single line.
14	Password	If set, the field is intended for entering a secure password that should not 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 spaced positions, or *combs*, as the value of **MaxLen**, and the text is laid out into those combs.
26	RichText	*(PDF 1.5)* If set, the value of this field should be represented as a rich text string (see "Rich Text Strings" on page 680). If the field has a value, the **RV** entry of the field dictionary (Table 8.71) specifies the rich text string.

The field's text is held in a text string (or, beginning with PDF 1.5, a stream) in the **V** (value) entry of the field dictionary. The contents of this text string or stream are used to construct an appearance stream for displaying the field, as described under "Variable Text" on page 677. The text is presented in a single style (font, size, color, and so forth), as specified by the **DA** (default appearance) string.

If the FileSelect flag *(PDF 1.4)* is set, the field functions as a *file-select control.* In this case, the field's text represents the pathname of a file whose contents are to be submitted as the field's value:

- For fields submitted in HTML Form format, the submission uses the MIME content type multipart/form-data, as described in Internet RFC 2045, *Multipurpose Internet Mail Extensions (MIME), Part One: Format of Internet Message Bodies* (see the Bibliography).
- For Forms Data Format (FDF) submission, the value of the **V** entry in the FDF field dictionary (see "FDF Fields" on page 717) is a file specification (Section 3.10, "File Specifications") identifying the selected file.
- 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
      /T (Silly prose)
      /DA (0 0 1 rg /Ti 12 Tf)
      /V (The quick brown fox ate the lazy mouse)
      /AP << /N 5 0 R >>
   >>
endobj
```

```
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
18	Combo	If set, the field is a combo box; if clear, the field is a list box.
19	Edit	If set, the combo box includes an editable text box as well as a drop-down list; if clear, it includes only a drop-down list. This flag is meaningful only if the Combo flag is set.

BIT POSITION	NAME	MEANING
20	Sort	If set, the field's option items should be sorted alphabetically. This flag is intended for use by form authoring tools, not by PDF viewer applications. Viewers should simply display the options in the order in which they occur in the **Opt** array (see Table 8.80).
22	MultiSelect	*(PDF 1.4)* If set, more than one of the field's option items may be selected simultaneously; if clear, no more than one item at a time may be selected.
23	DoNotSpellCheck	*(PDF 1.4)* If set, text entered in the field is not spell-checked. This flag is meaningful only if the Combo and Edit flags are both set.
27	CommitOnSelChange	*(PDF 1.5)* If set, the new value is committed as soon as a selection is made with the pointing device. This option enables applications to perform an action once a selection is made, without requiring the user to exit the field. If clear, the new value is not committed until the user 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, representing the zero-based indices in the **Opt** array of the currently selected option items. This entry is required when two or more elements in the **Opt** array have different names but the same export value or when the value of the choice field is an array. In other cases, the entry is permitted but not required. If the items identified by this entry differ from those in the **V** entry of the field dictionary (see below), the **V** entry takes precedence.

The **Opt** array specifies the list of options in the choice field, each of which is represented by a text string to be displayed on the screen. Each element of the **Opt** array contains either this text string by itself or a two-element array, whose second element is the text string and whose first element is a text string representing the export value to be used when exporting interactive form field data from the document.

The field dictionary's **V** (value) entry (see Table 8.69 on page 675) identifies the item or items currently selected in the choice field. If the field does not allow multiple selection—that is, if the MultiSelect flag *(PDF 1.4)* is not set—or if multiple selection is supported but only one item is currently selected, **V** is a text string representing the name of the selected item, as given in the field dictionary's **Opt** array. If multiple items are selected, **V** is an array of such strings. (For items represented in the **Opt** array by a two-element array, the name string is the second of the two array elements.) The default value of **V** is **null**, indicating that no item is currently selected.

Example 8.19 shows a typical choice field definition.

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 field may contain the additional entries listed in Table 8.81, as well as the standard entries described in Table 8.69. The field type (**FT**) is **Sig**, and the field value (**V**) is a *signature dictionary* containing the signature and specifying various attributes of the signature field (see Table 8.102).

Filling in (signing) the signature field entails updating at least the **V** entry and usually also the **AP** entry of the associated widget annotation. Exporting a signature field typically exports the **T**, **V**, and **AP** entries.

Like any other field, a signature field may actually be described by a widget annotation dictionary containing entries pertaining to an annotation as well as a field (see "Widget Annotations" on page 640). The annotation rectangle (**Rect**) in such a dictionary gives the position of the field on its page. Signature fields that are not intended to be visible should have an annotation rectangle that has zero height and width.

The appearance dictionary (**AP**) of a signature field's widget annotation defines the field's visual appearance on the page (see Section 8.4.4, "Appearance Streams"). Information about how Acrobat handles digital signature appearances is in the technical note *Digital Signature Appearances* (see the Bibliography).

TABLE 8.81 Additional entries specific to a signature field

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 pertinent to itself and ignore the others.*

TABLE 8.82 Entries in a signature field lock dictionary

KEY	TYPE	VALUE
Type	name	*(Optional)* The type of PDF object that this dictionary describes; if present, must be **SigFieldLock** for a signature field lock dictionary.
Action	name	*(Required)* A name which, in conjunction with **Fields**, indicates the set of fields that should be locked. Valid values are: All — All fields in the document Include — All fields specified in **Fields** Exclude — All fields except those specified in **Fields**
Fields	array	*(Required if the value of* ***Action*** *is Include or Exclude)* An array of text strings containing field names.

TABLE 8.83 Entries in a signature field seed value dictionary

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 first matching encoding should be used if it is available. If it is not available, a different encoding can be used.

KEY	TYPE	VALUE
DigestMethod	array	(*Optional; PDF 1.7*) An array of names indicating acceptable digest algorithms to use while signing. The valid values are **SHA1**, **SHA256**, **SHA384**, **SHA512** and **RIPEMD160**. The default value is implementation-specific. ***Note:*** *This property is only applicable if the digital credential signing contains RSA public/private keys. If it contains DSA public/ private key, the digest algorithm is always* ***SHA1*** *and this attribute is ignored.*
V	real	*(Optional)* The minimum required capability of the signature field seed value dictionary parser. A value of 1 specifies that the parser must be able to recognize all seed value dictionary entries specified in PDF 1.5. A value of 2 specifies that it must be able to recognize all seed value dictionary entries specified in PDF 1.7 and earlier. The **Ff** entry indicates whether this is a required constraint. ***Note:*** *The* PDF Reference fifth edition *(PDF 1.6) and earlier, erroneously indicates that the* **V** *entry is of type integer. This entry is of type real.*
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 an ordinary (non-author) signature (see Section 8.7, “Digital Signatures”). The values 1 through 3 are used for author signatures and correspond to the value of **P** in a **DocMDP** transform parameters dictionary (see Table 8.104). If this entry is not present or does not contain a **P** entry, no rules are defined regarding the type of signature or its permissions.

KEY	TYPE	VALUE
TimeStamp	dictionary	*(Optional; PDF 1.6)* A time stamp dictionary containing two entries: **URL** An ASCII string specifying the URL of a time-stamping server, providing a time stamp that is compliant with RFC 3161, *Internet X.509 Public Key Infrastructure Time-Stamp Protocol* (see the Bibliography). **Ff** An integer whose value is 1 (the signature is required to have a time stamp) or 0 (the signature is not required to have a time stamp). Default value: 0.
LegalAttestation	array	*(Optional; PDF 1.6)* An array of text strings specifying possible legal attestations (see Section 8.7.4, “Legal Content Attestations”). The value of the corresponding flag in the **Ff** entry indicates whether this is a required constraint.
AddRevInfo	boolean	(*Optional; PDF 1.7*) A flag indicating whether revocation checking should be carried out. If **AddRevInfo** is true, the viewer application performs the following additional tasks when signing the signature field: • 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 seed value dictionary

KEY	TYPE	VALUE
Type	name	*(Optional)* The type of PDF object that this dictionary describes; if present, must be **SVCert** for a certificate seed value dictionary.
Subject	array	*(Optional)* An array of byte strings containing DER-encoded X.509v3 certificates that are acceptable for signing. X.509v3 certificates are described in RFC 3280, *Internet X.509 Public Key Infrastructure, Certificate and Certificate Revocation List (CRL) Profile* (see the Bibliography). The value of the corresponding flag in the **Ff** entry indicates whether this is a required constraint.
SubjectDN	array of dictionaries	(*Optional; PDF 1.7*) An array of dictionaries, where each dictionary contains key value pairs, that specify the Subject Distinguished Name (DN) that must be present within the certificate for it to be acceptable for signing. The certificate must at a minimum contain all the attributes specified in the dictionary. That is, the certificate can contain additional attributes. The Subject Distinguished Name is described in RFC 3280 (see the Bibliography). The key can be any legal attribute identifier. Attribute names are typically of the form 'cn', 'o', '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 of ASCII strings, where each string specifies an acceptable key-usage extension that must be present in the signing certificate. Multiple strings specify a range of acceptable key-usage extensions. The key-usage extension is described in RFC 3280 (see the Bibliography). Each character in a string represents a key-usage type, where the order of the characters indicates the key-usage extension it represents. The first through ninth characters in the array, from left to right, represent the required value for the following key-usage extensions: 1 digitalSignature 4 dataEncipherment 7 cRLSign 2 non-Repudiation 5 keyAgreement 8 encipherOnly 3 keyEncipherment 6 keyCertSign 9 decipherOnly Any additional characters are ignored. Any missing characters or characters that are not one of the following values, should be set to 'X'. The following 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 example of such a string is (2.16.840.1.113733.1.7.1.1). This field is only applicable if the value of **Issuer** is not empty. The certificate policies extension is described in RFC 3280 (see the Bibliography). The value of the corresponding flag in the **Ff** entry indicates whether this is a required constraint.

KEY	TYPE	VALUE
URL	ASCII string	*(Optional)* A URL, the use for which is defined by the **URLType** entry.
URLType	Name	(*Optional; PDF 1.7*) A name indicating the usage of the **URL** entry. There are standard uses and there can be implementation-specific uses for this URL. The following value specifies a valid standard usage: **Browser** – The URL references content that should be displayed in a web browser to allow enrolling for a new credential if a matching credential is not found. The **Ff** attribute's URL bit is ignored for this usage. The following value specifies a valid implementation-specific usage, defined for use by Adobe Systems: **ASSP** – The URL references a signature web service that can be used for server-based signing. If the **Ff** attribute's URL bit indicates that this is a required constraint, this implies that the credential used when signing must come from this server. Third parties can extend the use of this attribute with their own attribute values, 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 fields to a specified uniform resource locator (URL), presumably the address of a Web server that will process them and send back a response.
- *Reset-form actions* reset selected interactive form fields to their default values.
- *Import-data actions* import Forms Data Format (FDF) data into the document's interactive form from a specified file.

- *JavaScript actions (PDF 1.3)* cause a script to be compiled and executed by the JavaScript interpreter.

Submit-Form Actions

A *submit-form action* transmits the names and values of selected interactive form fields to a specified uniform resource locator (URL), presumably the address of a Web server that will process them and send back a response. Table 8.85 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). Table 8.86 shows the meanings of the flags; all undefined flag bits are reserved and must be set to 0.

TABLE 8.85 Additional entries specific to a submit-form action

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 with no values may also be excluded, depending on the setting of the IncludeNoValueFields flag; see Table 8.86.) See the text following Table 8.86 for further discussion.
Flags	integer	*(Optional; inheritable)* A set of flags specifying various characteristics of the action (see Table 8.86). Default value: 0.

TABLE 8.86 Flags for submit-form actions

BIT POSITION	NAME	MEANING
1	Include/Exclude	If clear, the **Fields** array (see Table 8.85) specifies which fields to include in the submission. (All descendants of the specified fields in the field hierarchy are submitted as well.) If set, the **Fields** array tells which fields to exclude. All fields in the document's interactive form are submitted *except* those listed in the **Fields** array and those whose NoExport flag (see Table 8.70 on page 676) is set.
2	IncludeNoValueFields	If set, all fields designated by the **Fields** array and the Include/Exclude flag are submitted, regardless of whether they have a value (**V** entry in the field dictionary). For fields without a value, only the field name is transmitted. If clear, fields without a value are not submitted.
3	ExportFormat	Meaningful only if the SubmitPDF and XFDF flags are clear. If set, 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 x=*xval*&y=*yval* This flag is meaningful only when the ExportFormat flag is set. If ExportFormat is clear, this flag must also be clear.

BIT POSITION	NAME	MEANING
6	XFDF	*(PDF 1.4)* Meaningful only if the SubmitPDF flags are clear. If set, field names and values are submitted as XFDF.
7	IncludeAppendSaves	*(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). If set, the submitted FDF file includes the contents of all incremental updates to the underlying PDF document, as contained in the **Differences** entry in the FDF dictionary (see Table 8.93 on page 714). If clear, the incremental updates are not included.
8	IncludeAnnotations	*(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). If set, the submitted FDF file includes all markup annotations in the underlying PDF document (see "Markup Annotations" on page 616). If clear, markup annotations are not included.
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 displayed in the title bar of the annotation's pop-up window and is assumed to represent the name of the user authoring the annotation.) This allows multiple users to collaborate in annotating a single remote PDF document without affecting one another's annotations.

BIT POSITION	**NAME**	**MEANING**
12	ExclFKey	*(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). If set, the submitted FDF excludes the **F** entry.
14	EmbedForm	*(PDF 1.5)* Meaningful only when the form is being submitted in Forms Data Format (that is, when both the XFDF and ExportFormat flags are clear). If set, the **F** entry of the submitted FDF is a file specification containing an embedded file stream representing the PDF file from which the FDF is being submitted.

The set of fields whose names and values are to be submitted is defined by the **Fields** array in the action dictionary (Table 8.85) together with the Include/Exclude and IncludeNoValueFields flags in the **Flags** entry (Table 8.86). Each element of the **Fields** array identifies an interactive form field, either by an indirect reference to its field dictionary or *(PDF 1.3)* by its fully qualified field name (see "Field Names" on page 676). If the Include/Exclude flag is clear, the submission consists of all fields listed in the **Fields** array, along with any descendants of those 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.
- XFDF, a version of FDF based on XML. XFDF is described in the Adobe technical note *XML Forms Data Format Specification, Version 2.0.* XML is described in the W3C document *Extensible Markup Language (XML) 1.1*)
- PDF (in this case, the entire document is submitted rather than individual fields and values).

The name submitted for each field is its fully qualified name (see “Field Names” on page 676), and the value is specified by the **V** entry in its field dictionary.

Note: *For pushbutton fields submitted in FDF, the value submitted is that of the* ***AP*** *entry in the field’s widget annotation dictionary. If the submit-form action dictionary contains no* ***Fields*** *entry, such pushbutton fields are not submitted at all.*

Fields with no value (that is, whose field dictionary does not contain a **V** entry) are ordinarily not included in the submission. The submit-form action’s Include-NoValueFields flag can override this behavior. If this flag is set, such valueless fields are included in the submission by name only, with no associated value.

Reset-Form Actions

A *reset-form action* resets selected interactive form fields to their default values; that is, it sets the value of the **V** entry in the field dictionary to that of the **DV** entry (see Table 8.69 on page 675). If no default value is defined for a field, its **V** entry is removed. For fields that can have no value (such as pushbuttons), the action has 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.