INTERNATIONAL STANDARD

ISO 19444-1

First edition
2016-12-15

Document management — XML Forms Data Format —

Part 1: Use of ISO 32000-2 (XFDF 3.0)

Gestion de documents — Format de Données des Formulaires XML — Partie 1: Utilisation de l'ISO 32000-2 (XFDF 3.0)

Reference number
ISO 19444-1:2016(E)

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see the following URL: www.iso.org/iso/foreword.html.

The committee responsible for this document is ISO/TC 171, *Document management applications*, Subcommittee SC 2, *Document file formats, EDMS systems and authenticity of information*.

A list of all parts in the ISO 19444 series can be found on the ISO website.

Introduction

This document describes the XML Forms Data Format, which is used to represent form data from PDF (ISO 32000-2) in an XML tagset.

This format is derived from the Forms Data Format in PDF and is intended to be a more interchangeable format for forms data.

Document management — XML Forms Data Format —

Part 1:
Use of ISO 32000-2 (XFDF 3.0)

1 Scope

This document specifies an XML format for representing forms data and annotations in the Portable Document Format, ISO 32000-2 (PDF 2.0).

This document does not change or add any definitions for any components of ISO 32000-2.

2 Normative references

There are no normative references in this document.

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO 32000-2 apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— IEC Electropedia: available at http://www.electropedia.org/

— ISO Online browsing platform: available at http://www.iso.org/obp

4 Notation

All XFDF element names, XFDF attributes names and examples (XFDF and PDF) are written in fixed width font. All inline PDF and FDF operators, keywords and the names of keys in PDF dictionaries are written in bold font. All inline words that denote operands of PDF and FDF operators or values of PDF dictionary keys are written in italic sans serif font.

Ellipses (...) are used within XFDF examples to indicate omitted detail.

5 Introduction to XFDF

5.1 General

XML Forms Data Format (XFDF) is a format for representing forms data and annotations in a PDF document. This specification describes XFDF compatible with ISO 32000-2. XFDF is the XML version of Forms Data Format (FDF), a simplified version of PDF for representing forms data and annotations.

5.2 Forms data and annotations

Form fields in a PDF document may include many interactive elements (see ISO 32000-2:—, 12.7). Figure 1 shows a sample PDF which utilizes edit boxes, buttons, and radio buttons:

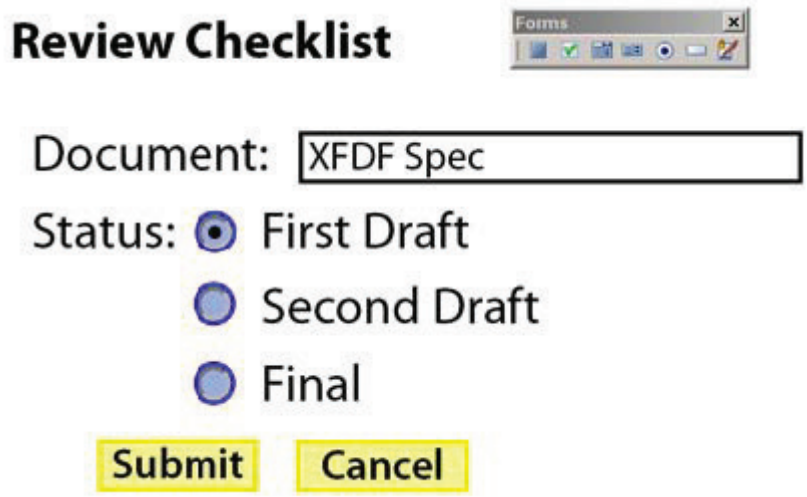

Figure 1 — Sample PDF utilizing edit boxes, buttons and radio buttons

The XFDF exported from this PDF document might look like the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
<xfdf xmlns="http://ns.adobe.com/xfdf/" xml:space="preserve">
     <f href="Checklist.pdf"/>
     <ids original="7A0631678ED475F0898815F0A818CFA1"
     modified="BEF7724317B311718E8675B677EF9B4E"/>
     <fields>
     ...
     </fields>
</xfdf>
```

XFDF is often used as the format to send and receive PDF form data from a server. Form data are submitted to a server, modifications are made and sent back; the new form data are imported into the interactive PDF form. XFDF is also often used as a format to export form data to stand-alone files that can be stored, transmitted electronically, and imported back into the corresponding PDF interactive form.

Annotations (see ISO 32000-2:—, 12.5) are attached to a PDF document. Figure 2 shows an example PDF which includes text notes, highlights, stamps, and file attachments:

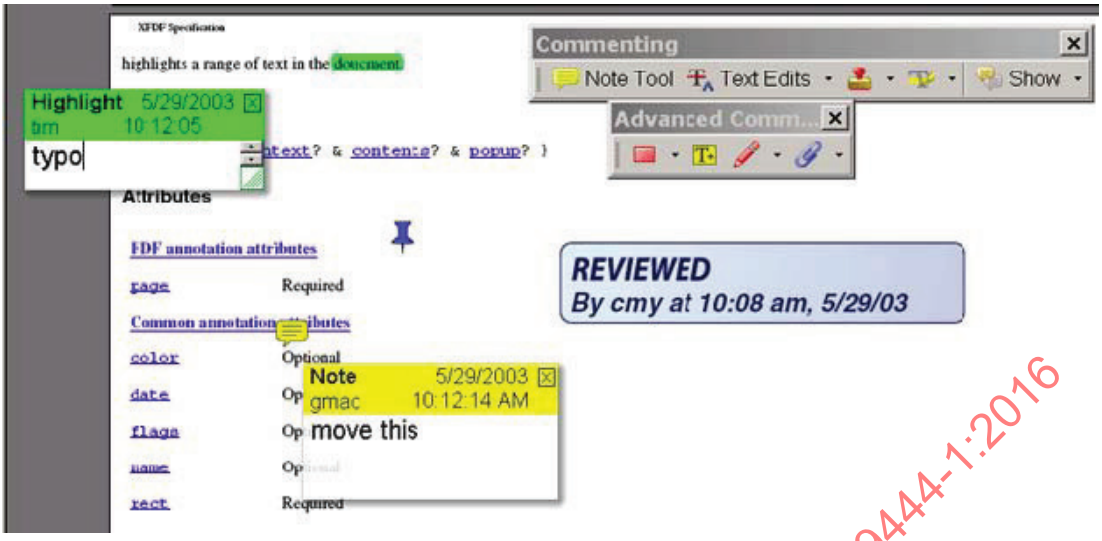

Figure 2 — Example PDF shown with text notes, highlights, stamps and file attachments

5.3 How to use this specification

This specification documents the correspondence between an XFDF element or attribute and its equivalent PDF dictionary and key. A short description is provided for each element and attribute; for complete information, see the description of the corresponding dictionary and key in ISO 32000-2. There are also a few attributes that do not correspond to a specific PDF dictionary and key.

5.4 PDF, FDF and XFDF

5.4.1 General

PDF, FDF, and XFDF are related specifications with PDF as the parent format for representing documents, including interactive forms and annotations. FDF and XFDF contain the subset of a PDF document that describes interactive forms and annotations. Complete information on PDF and FDF may be found in ISO 32000-2.

FDF is a simplified version of PDF. PDF and FDF represent information with a key/value pair, also referred to as an entry. This example shows the **T** and **V** keys with values enclosed in parentheses:

```
/T (Street) /V (345 Park Ave.)
```

XFDF, on the other hand, represents an entry with an XML element/content or attribute/value pair, as shown in the corresponding XFDF:

```
<field name="Street">
<value>345 Park Ave.</value>
</field>
```

XFDF implements a subset of FDF containing forms and annotations. There are XFDF equivalents for the **Annots**, **Fields**, **F**, and **ID** keys of the FDF dictionary. There are no XFDF equivalents for the other entries in the FDF dictionary such as the **Status**, **Encoding**, **JavaScript**, **EmbeddedFDFs**, **Differences**, **Target**, and **Pages** keys.

XFDF conforms to the XML standard (see W3C, Extensible Markup Language 1.0).

In the simplest case, an XFDF element or attribute maps directly to a key in a particular dictionary of PDF. For example, the `creationdate` attribute is documented as corresponding to the **CreationDate** key

in the markup annotation dictionary. This specification provides a description of the `creationdate` attribute, but more detailed information can be found in ISO 32000-2.

An example `creationdate` attribute in XFDF:

```
creationdate="D:20030425095243-07'00"
```

An equivalent entry in PDF or FDF would look like:

```
/CreationDate (D:20030425095243-07'00)
```

In the example above, the underlying data stored within XFDF or PDF is identical. However, in other cases, the name and value differ. For example, the `flags` attribute corresponds to the **F** key in the annotation dictionary. The value of the `flags` attribute is a comma separated list of the descriptive names of the flags, while the value of the **F** key is an integer with each bit representing a flag.

This is the XFDF flags attribute:

```
flags="print,nozoom,norotate"
```

This is the equivalent F entry in PDF or FDF:

```
/F 28
```

Finally, an element with multiple attributes can map to a single key with multiple values. The `ids` element in XFDF has attributes original and modified that map to the **ID** key in the FDF dictionary.

Below is an example `ids` element in XFDF:

```
<ids original="7A0631678ED475F0898815F0A818CFA1"
modified="BEF7724317B311718E8675B677EF9B4E" />
```

This is the corresponding ID entry in FDF:

```
/ID [<7a0631678ed475f0898815f0a818cfa1><bef7724317b311718e8675b677ef9b4e>]
```

5.4.2 Sample form in FDF and XFDF

Both FDF and XFDF for forms contain the same information: field name and value. In this FDF example, with line returns added for readability, the **Fields** key contains two fields named Street and City:

```
%FDF-1.2
%âãÏÓ
1 0 obj
<< /FDF
        << /F (Document.pdf)
              /ID [ <7a0631678ed475f0898815f0a818cfa1>
                    <bef7724317b311718e8675b677ef9b4e> ]
       /Fields [
                     << /T (Street)
                        /V (345 Park Ave.) >>
                     << /T (City) /V (San Jose) >>
       ] >>
>>
endobj
trailer
<< /Root 1 0 R >>
%%EOF
```

This is the XFDF version of the same form fields. The `fields` element contains two field elements with attribute name set to `Street` and `City`:

```
<?xml version="1.0" encoding="UTF-8"?>
<xfdf xmlns="http://ns.adobe.com/xfdf/" xml:space="preserve">
       <f href="Document.pdf"/>
       <ids original="7A0631678ED475F0898815F0A818CFA1"
modified="BEF7724317B311718E8675B677EF9B4E"/>
       <fields>
              <field name="Street">
                     <value>345 Park Ave.</value>
              </field>
              <field name="City">
                     <value>San Jose</value>
              </field>
       </fields>
</xfdf>
```

5.4.3 Sample annotation in FDF and XFDF

XFDF and FDF contain similar information but XFDF is represented in the XML format. This is a snippet of an FDF file containing a note annotation (line breaks added for readability):

```
%FDF-1.2
%âãÏÓ
1 0 obj
<< /FDF
   << /F (/C/Samples/Document.pdf)
      /ID [ <7a0631678ed475f0898815f0a818cfa1>
            <bef7724317b311718e8675b677ef9b4e> ]
      /Annots [4 0 R 3 0 R] >>
>>
endobj

3 0 obj
<< ... >>
endobj

4 0 obj
<< /F 28
   /Page 0
   ...
   /Type /Annot
   /Subj (Note)
   /Rect [271.850464 690.255371 291.850464 708.255371]
   /CreationDate (D:20030425095243-07'00')
   /NM (apYVRecPEj75sYIwSxME7C)
   ...
   /Subtype /Text
   ... >>
endobj

trailer
<< /Root 1 0 R >>
%%EOF
```

This is the same data in XFDF format:

```
<?xml version="1.0" encoding="UTF-8"?>
<xfdf xmlns="http://ns.adobe.com/xfdf/" xml:space="preserve">
 <f href="Document.pdf"/>
 <ids original="7A0631678ED475F0898815F0A818CFA1" modified="BEF-
7724317B311718E8675B677EF9B4E"
/>
<annots>
   <text flags="print,nozoom,norotate" page="0" subject="Note" rect="271.85046
4,690.255371,291.850464,708.255371"
```



```
creationdate="D:20030425095243-07'00'" name="apYVRecPEj75sYIwSxME7C" ...
>
      ...
      <popup .../>
    </text>
  </annots>
</xfdf>
```

5.5 Writing XFDF

5.5.1 General

This clause describes XML implementation details specific to XFDF.

5.5.2 Encoding and Namespace

The encoding in the XFDF file shall be UTF-8. Each XFDF file shall begin with the line:

```
<?xml version="1.0" encoding="UTF-8"?>
```

The namespace for XFDF shall be:

http://ns.adobe.com/xfdf/

The XML shall always specify that space is preserved in the XFDF files as follows:

```
xml:space="preserve"
```

Therefore, an XFDF document shall always begin with the following two lines:

```
<?xml version="1.0" encoding="UTF-8"?>
<xfdf xmlns="http://ns.adobe.com/xfdf/" xml:space="preserve">
```

5.6 Understanding forms

5.6.1 General

An XFDF file with form data contains form field names and values. When importing XFDF into a PDF processor, the target PDF file must already contain the form fields. Importing XFDF updates the form field values in the PDF file. Exporting to XFDF puts the current value of the field in the `value` element.

Using XFDF, it is not possible to create a new form field in a PDF document, or change anything other than the value of an existing form field.

5.6.2 Simple XFDF form

Figure 3 depicts a simple PDF document representing an address label containing text box form fields named *Name*, *Street* and *CityState*.

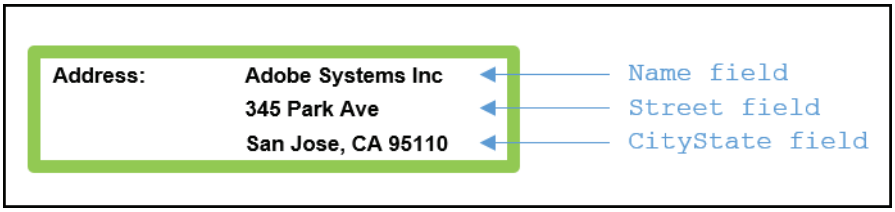

Figure 3 — PDF representing an address label showing example form field names

In the example below, the `href` attribute on the `f` element points to the PDF document that contains the form fields. The `ids` element's original attribute contains a permanent identifier for the file, and the modified attribute contains an identifier that changes with each modification to the file. The `fields` element contains the three form fields and their respective values.

```
<?xml version="1.0" encoding="UTF-8"?>
<xfdf xmlns="http://ns.adobe.com/xfdf/" xml:space="preserve">
 <f href="samples/AddressLabel.pdf"/>
 <ids original="7A0631678ED475F0898815F0A818CFA1"
modified="BEF7724317B311718E8675B677EF9B4E"/>
<fields>
   <field name="Name">
      <value>Adobe Systems, Inc.</value>
   </field>
   <field name="Street">
      <value>345 Park Ave.</value>
   </field>
   <field name="CityState">
      <value>San Jose, CA 95110</value>
   </field>
</fields>
</xfdf>
```


5.6.3 Hierarchical XFDF form

Hierarchical form fields are commonly represented using a dot notation. If Name, Street and CityState are part of an Address, the fields are named:

```
Address.Name Address.Street Address.CityState
```

Figure 4 show the same PDF rendition as Figure 3, but the field names are changed to hierarchical form fields.

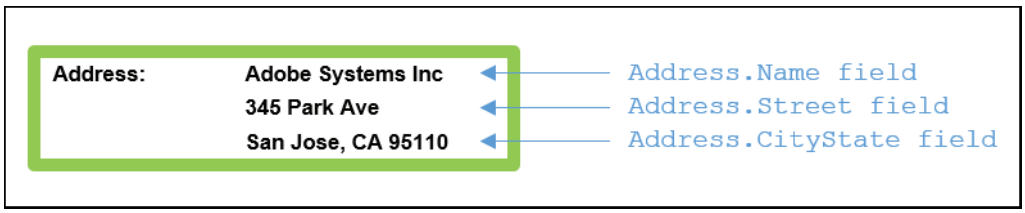

Figure 4 — PDF representing an address label showing example hierarchical form field names

In XFDF exported from this PDF file, hierarchical form fields are represented using nested field elements. The Address field contains the `Name`, `Street` and `CityState` fields:

```
<?xml version="1.0" encoding="UTF-8"?>
<xfdf xmlns="http://ns.adobe.com/xfdf/" xml:space="preserve">
 <fields>
   <field name="Address">
      <field name="Name">
         <value>Adobe Systems, Inc.</value>
      </field>
      <field name="Street">
         <value>345 Park Ave.</value>
      </field>
      <field name="CityState">
         <value>San Jose, CA 95110</value>
      </field>
   </field>
 </fields>
</xfdf>
```

5.7 Understanding annotations

5.7.1 General

XFDF annotations contain full information to recreate the annotation in a PDF document, including size and position on the page, the open or closed state of the annotation, colour, and attached comments. Unlike forms, a new annotation can be created when XFDF is imported into a PDF file. However, this means that the XFDF for annotations is more complex than for forms.

Markup and **Popup** annotations are represented in XFDF; there are only eight annotations that are not represented in XFDF. Each annotation is represented by an element: for example, a **Text** annotation is represented by the `text` element, and a **Polygon** annotation is represented by the `polygon` element. This table lists annotations that are supported and unsupported by XFDF.

Table 1 — List of supported and unsupported annotations

Supported annotation types	Unsupported annotations
`Text`	`Movie`
`FreeText`	`Widget`
`Line`	`Screen`
`Square`	`PrinterMark`
`Circle`	`TrapNet`
`Polygon`	`RichMedia` (PDF 2.0)
`Polyline`	`3D` (PDF 2.0)
`Highlight`	`Watermark` (PDF 2.0)
`Underline`	
`Squiggly`	
`StrikeOut`	
`Stamp`	
`Caret`	
`Ink`	

Table 1 *(continued)*

Supported annotation types	Unsupported annotations
`Popup`	
`FileAttachment`	
`Sound`	
`Link`	
`Redact`	
`Projection` (PDF 2.0)	

5.7.2 Simple XFDF annotation

Figure 5 shows an example where a stamp annotation has been applied to a page in a PDF file.

Figure 5 — Stamp annotation applied to a PDF file

In the example below, the `href` attribute on the `f` element contains the name of the PDF file that exported the annotations. The `ids` element's original attribute contains a permanent identifier for the file, and the modified attribute contains an identifier that changes with each modification to the file.

Next is the `annots` element, which contains all annotations in the document. In this case, there is only one `stamp` annotation. In contrast to forms, annotations have many attributes, such as `color` or `title`, that can be modified and imported back into the PDF file to change the look of the annotation.

The `stamp` element contains a popup element which corresponds to the popup window for adding comments that is attached to the annotation. In this example, the popup window is empty and closed (`open="no"`).

```
<?xml version="1.0" encoding="UTF-8"?>
<xfdf xmlns="http://ns.adobe.com/xfdf/" xml:space="preserve">
 <f href="SimpleAnnot.pdf"/>
 <ids original="7A0631678ED475F0898815F0A818CFA1" modified="BEF-
7724317B311718E8675B677EF9B4E"/>
 <annots>
   <stamp flags="print" page="0" subject="Approved" rect="54.987381,671.039063
,216.486893,718.539551"
creationdate="D:20030528192526-07'00'" name="jNrKlQf-J0kz3Y3a0cPjzA" icon="S-
BApproved" color="#FF0000" date="D:20030528192529-07'00"
title="cmy">
      <popup flags="print,nozoom,norotate" page="0" rect="612.000000,619.06597
9,792.000000,739.065979"
open="no"/>
   </stamp>
 </annots>
</xfdf>
```

5.7.3 Annotation with popup text

Figure 6 shows an example of the rubber stamp annotation with an open popup note with text.

Figure 6 — Example of a rubber stamp annotation with an open popup note

In the exported XFDF for the stamp element, the text of the popup is contained in a `contents-richtext` element which contains elements that conform to a subset of the XFA text specification. These are commonly referred to as rich text strings. For more information on rich text strings, see 5.8.3. Here is the new stamp element with some attributes removed for readability:

```
<?xml version="1.0" encoding="UTF-8"?>
<xfdf xmlns="http://ns.adobe.com/xfdf/" xml:space="preserve">
 <stamp page="0" subject="Approved" icon="SBApproved" title="cmy">
   <contents-richtext>
      <body xmlns="http://www.w3.org/1999/xhtml" xmlns:xfa="http://www.xfa.org/
schema/xfa-data/1.0/" xfa:APIVersion="3.1.9078.0" xfa:spec="3.1">
         <p>
            <span style="font-size:10.0pt">Final and ready for publishing.</span>
         </p>
      </body>
   </contents-richtext>
   <popup page="0"
rect="264.527802,648.970642,369.027802,715.470642"
open="yes"/>
 </stamp>
</xfdf>
```

5.7.4 Annotation with comment

Annotations can have comments attached to them. In Figure 7, the rubber stamp annotation has one comment.

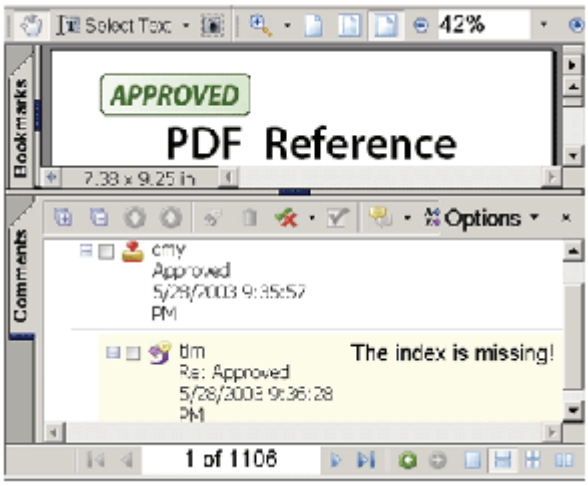

Figure 7 — Example of a rubber stamp annotation with one comment

The example below shows the `annots` element exported from the PDF file (attributes have been removed to improve readability). The comment is contained in the `text` element which is the second child of the `annots` element and follows the `stamp` element. The `text` element represents a comment because the value of the `inreplyto` attribute on text is identical to the value of the `name` attribute on stamp. The text of the annotation is contained in the `contents-richtext` element which is described in 5.8.3.

```
<?xml version="1.0" encoding="UTF-8"?>
<xfdf xmlns="http://ns.adobe.com/xfdf/" xml:space="preserve">
 <annots>
   <stamp subject="Approved" name="HLjJ_qj5BC9dU1yKDfFD6D" icon="SBApproved"
title="cmy">
      <popup open="no"/>
   </stamp>
   <text subject="Re: Approved" name="miAYuQ7A9JvIb3mFNkLjzC"
inreplyto="HLjJ_qj5BC9dU1yKDfFD6D" icon="Comment" title="tim">
      <contents-richtext>
         <body xmlns="http://www.w3.org/1999/xhtml" xmlns:xfa="http://www.
xfa.org/schema/xfa-data/1.0/"
xfa:APIVersion="3.1.9078.0" xfa:spec="3.1">
            <p>
               <span style="font-size:10.0pt">The index is missing!</span>
            </p>
         </body>
      </contents-richtext>
      <popup open="no"/>
   </text>
 </annots>
</xfdf>
```

5.8 Implementation notes

5.8.1 General

This clause contains useful information regarding processing of the XFDF, as well as understanding the notation used in the XFDF Reference section later.

5.8.2 String encoding conventions

XML requires that all content be in some particular character encoding. Much of PDF also has this requirement, but there are some strings in PDF for which the encoding is not known. In PDF these strings are designated as "string" and are effectively byte strings without any particular character interpretation.

The following convention is recommended for transforming these strings between PDF and XML.

— Use ISO-Latin1 as the assumed encoding of the bytes in the PDF. For example, for the link annotation, this applies to the Named Destination name and the file `OriginalName`.

— Escape any characters that are XML reserved or not legal code points in ISO-Latin1. Specifically, the escaping mechanism is:

  — If char is 0x0A, 0x0D or 0x09, emit

 or 	 respectively.

  — Else if char<0x20, emit escaped octal code (just like escaped sequences in PDF literal string). For example, code point 0x07 is emitted as \007.

  — Else if char is 0x22, 0x26, 0x3C, 0x3E (XML delimiters), emit " & < or > respectively.

5.8.3 Rich text strings

Beginning with PDF 1.5, the text contents of variable text form fields and markup annotations can include formatting or style information. These rich text strings conform to a subset of the XFA Text Specification, which is itself a subset of the XHTML 1.0 specification, augmented with a restricted set of CSS2 style attributes. Rich text strings are fully described in ISO 32000-2.

Figure 8 shows a text field form that has a value formatted as rich text.

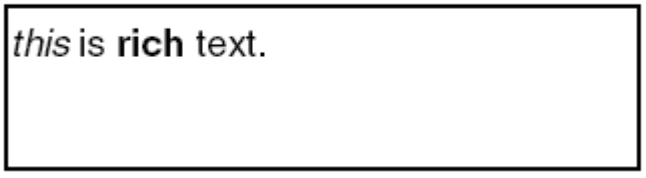

Figure 8 — Example of rich text

For example, the following text field form has a value formatted as rich text:

```
<?xml version="1.0" encoding="UTF-8"?>
<xfdf xmlns="http://ns.adobe.com/xfdf/" xml:space="preserve">
 <field name="myfield">
   <value-richtext>
      <body xmlns="http://www.w3.org/1999/xhtml" xmlns:xfa="http://www.xfa.
org/schema/xfa-data/1.0/" xfa:APIVersion="3.1.9078.0" xfa:spec="3.1">
```

```
            <p>
               <span style="font-size:10.0pt">
                  <i>this</i>is<b>rich</b>text.</span>
            </p>
         </body>
      </value-richtext>
   </field>
</xfdf>
```

5.8.4 XML content model syntax

5.8.4.1 Syntax overview

In the element reference (see 6.2), a content model section is provided for each element. The content model defines the elements or types of text strings that can be contained by the element. For example, the content model for the xfdf element is:

```
(f? & ids? & fields? & annots?)
```

The content model is written using the symbols described in Table 2.

Table 2 — XML content model syntax

Symbol	Description
(	begin group
)	end group
,	followed by
&	and
|	or
?	0 or 1
+	1 or more
*	0 or more

The following are a few examples of content models.

5.8.4.2 Example 1

If element `lunch` can contain `salad` or `soup`, followed by `sandwich`, followed by an optional `dessert`, the content model is:

```
(salad | soup), sandwich, dessert?
```

The following are valid lunch menus:

```
<lunch><salad/><sandwich/></lunch>
<lunch><soup/><sandwich/><dessert/></lunch>
```

However, the following is not a valid lunch because you cannot have both `salad` and `soup`:

```
<lunch><salad/><soup/><sandwich/></lunch>
```

The following is not valid because you must have `salad` or `soup` and you cannot have two `dessert` elements:

```
<lunch><sandwich/><dessert/><dessert/></lunch>
```

5.8.4.3 Example 2

If element `sandwich` can contain, in any order, an optional `tomato` and optional `lettuce` element, the content model is:

```
(tomato? & lettuce?)
```


These are valid `sandwich` elements:

```
<sandwich><tomato/><lettuce/></sandwich>
<sandwich><lettuce/><tomato/></sandwich>
<sandwich/>
<sandwich><lettuce/></sandwich>
<sandwich><tomato/></sandwich>
```

This `sandwich` is not valid because it contains an extra `tomato`:

```
<sandwich><tomato/><lettuce/><tomato/><sandwich/>
```

6 XFDF reference

6.1 General

The following clauses define a written schema for the elements and attributes that shall be used to represent an XFDF document. A conformant XFDF document shall contain only elements defined within this document. Each element shall contain only those elements defined within its content model and only those attributes specified for that respective element. All attributes specified as required shall be present if that element is used in the document and shall conform to the data types identified for that attribute.

6.2 XFDF elements

6.2.1 Elements

This clause describes the top level `xfdf` element and two of its children.

6.2.2 xfdf

The `xfdf` element is the top level element in an XFDF document.

6.2.2.1 Content model

```
(f? & ids? & fields? & annots?)
```

6.2.2.2 Attributes

`xml:space` *Required*. Value must be `preserve`. This attribute in the xml namespace indicates that whitespace is preserved.

6.2.3 f

The `f` element is a child of the `xfdf` element and corresponds to the **F** key in the FDF dictionary. Specifies the source file or target file: the PDF document that this XFDF file was exported from or is intended to be imported into.

6.2.3.1 Content model

Empty

6.2.3.2 Attributes

`href` *Required*. File specification (see ISO 32000-2:—, 7.11, File Specifications) pointing to the source file or target file.

6.2.4 ids

The `ids` element is a child of the `xfdf` element. The `ids` element corresponds to the **ID** Key in the FDF dictionary. The two attributes are file identifiers for the source or target file designated by the `f` element, taken from the **ID** entry in the PDF file's trailer dictionary.

6.2.4.1 Content model

Empty

6.2.4.2 Attributes

`original` *Required*. This attribute corresponds to the permanent identifier which is based on the contents of the file at the time it was originally created. This value does not change when the file is incrementally updated.

The value shall be a hexadecimal number.

NOTE: A common value for this is an MD5 checksum.

`modified` *Required*. The `modified` attribute contains a unique identifier for the modified version of the PDF and corresponding XFDF document. The `modified` attribute corresponds to the changing identifier that is based on the file's contents at the time it was last updated.

The value shall be a hexadecimal number.

NOTE: A common value for this is an MD5 checksum.

6.3 Form field elements

6.3.1 Fields

6.3.1.1 Overview

The `fields` element is a child of the `xfdf` element and is the container for form `field` elements. The `fields` element corresponds to the **Fields** key in the FDF dictionary.

6.3.1.2 Content model

*field**

6.3.1.3 Attributes

None

6.3.2 Field

6.3.2.1 Overview

The `field` element is a child of the `fields` and `field` elements. The field element corresponds to a form field.

6.3.2.2 Content model

(field | value* | (value? & value-richtext?))*

6.3.2.3 Attributes

`name` — *Required*. The name attribute corresponds to the **T** key in the FDF field dictionary. In a hierarchical form field, the name is the partial field name.

6.3.2.4 Details

Hierarchical fields are represented by nesting `field` elements. In PDF, hierarchical fields are named with a dot notation: *phone.work* and *phone.home*. In XFDF, these are represented as:

```
<field name="phone">
   <field name="work"/>
   <field name="home"/>
</field>
```

6.3.3 value

6.3.3.1 Overview

The `value` element is a child of the `field` element and contains the `field's` value, whose format may vary depending on the field type. The value corresponds to the **V** key in the FDF field dictionary.

A newline character in a PDF multi-line text field becomes a single line feed character in the contents of the `value` element.

Signature fields do not export a value.

6.3.3.2 Content model

Text string

6.3.3.3 Attributes

None

6.3.4 value-richtext

6.3.4.1 Overview

The `value-richtext` element is a child of the `field` element and contains the field's value formatted as a rich text string. This value corresponds to the **RV** key in the variable text field dictionary.

6.3.4.2 Content model

Text string or rich text string.

NOTE See 5.8.3.

6.3.4.3 Attributes

None

6.4 Annotation elements

6.4.1 annots

6.4.1.1 Overview

The annots element is a child of the xfdf element and serves as a container for annotation elements. The annots element corresponds to the **Annots** key in the FDF dictionary.

6.4.1.2 Content model

```
(text | highlight | underline | strikeout | squiggly | line | circle | square | caret
| polygon | polyline | stamp | ink | freetext | fileattachment | sound | link | redact
| projection)*
```

6.4.1.3 Attributes

None

6.4.2 text

6.4.2.1 Overview

The `text` element is a child of the `annots` element and corresponds to a text annotation. A text annotation represents a "sticky note" attached to a page in the PDF document.

6.4.2.2 Content model

(contents-richtext? & contents? & popup?)

6.4.2.3 Attributes

FDF annotation attributes

`page`	*Required*

Common annotation attributes

`color`	*Optional*
`date`	*Optional*
`flags`	*Optional*
`name`	*Optional*

`rect` *Required*

`title` *Optional*

Markup annotation attributes

`creationdate` *Optional*

`opacity` *Optional*

`subject` *Optional*

Text annotation attributes

`icon` *Optional*

`state` *Optional*

`statemodel` *Optional*

`inreplyto` *Optional*

`replyType` *Optional*

6.4.3 highlight

6.4.3.1 Overview

The `highlight` element is a child of the `annots` element and corresponds to the highlight Text annotation. A highlight annotation highlights a range of text in the document.

6.4.3.2 Content model

(contents-richtext? & contents? & popup?)

6.4.3.3 Attributes

FDF annotation attributes

`page` *Required*

Common annotation attributes

`color` *Optional*

`date` *Optional*

`flags` *Optional*

`name` *Optional*

`rect` *Required*

`title` *Optional*

Markup annotation attributes

`creationdate` *Optional*

`opacity`	*Optional*
`subject`	*Optional*

Text annotation attributes

`coords`	*Required*

6.4.4 underline

6.4.4.1 Overview

The `underline` element is a child of the `annots` element and corresponds to the Underline Text Markup annotation. An Underline annotation appears as an underline in the text of the document.

6.4.4.2 Content model

```
(contents-richtext? & contents? & popup?)
```

6.4.4.3 Attributes

FDF annotation attributes

`page`	*Required*

Common annotation attributes

`color`	*Optional*
`date`	*Optional*
`flags`	*Optional*
`name`	*Optional*
`rect`	*Required*
`title`	*Optional*

Markup annotation attributes

`creationdate`	*Optional*
`opacity`	*Optional*
`subject`	*Optional*

Text annotation attributes

`coords`	*Required*
`intent`	*Optional*

6.4.5 strikeout

6.4.5.1 Overview

The `strikeout` element is a child of the `annots` elements and corresponds to the Strikeout Text Markup annotation. A Strikeout annotation appears as a strikeout in the text of the document.

6.4.5.2 Content model

(contents-richtext? & contents? & popup?)

6.4.5.3 Attributes

FDF annotation attributes

`page`	*Required*

Common annotation attributes

`color`	*Optional*
`date`	*Optional*
`flags`	*Optional*
`name`	*Optional*
`rect`	*Required*
`title`	*Optional*

Markup annotation attributes

`creationdate`	*Optional*
`opacity`	*Optional*
`subject`	*Optional*

Text annotation attributes

`coords`	*Required*

6.4.6 squiggly

6.4.6.1 Overview

The `squiggly` element is a child of the `annots` element and corresponds to the Squiggly Text Markup annotation. The Squiggly annotation appears as a jagged underline in the text of a document.

6.4.6.2 Content model

(contents-richtext? & contents? & popup?)

6.4.6.3 Attributes

FDF annotation attributes

`page`	*Required*

Common annotation attributes

`color`	*Optional*
`date`	*Optional*

flags	*Optional*
name	*Optional*
rect	*Required*
title	*Optional*

Markup annotation attributes

creationdate	*Optional*
opacity	*Optional*
subject	*Optional*

Text annotation attributes

coords	*Required*

6.4.7 line

6.4.7.1 Overview

The `line` element is a child of the `annots` element and corresponds to the Line annotation. A Line annotation displays a single straight line on the page.

6.4.7.2 Content model

```
(contents-richtext? & contents? & popup?)
```

6.4.7.3 Attributes

FDF annotation attributes

page	*Required*

Common annotation attributes

color	*Optional*
date	*Optional*
flags	*Optional*
name	*Optional*
rect	*Required*
title	*Optional*

Markup annotation attributes

creationdate	*Optional*
opacity	*Optional*
subject	*Optional*

Text annotation attributes

`start`	*Required*
`end`	*Required*
`head`	*Optional*
`tail`	*Optional*
`interior-color`	*Optional*
`leaderLength`	*Optional*
`leaderExtended`	*Optional*
`caption`	*Optional*
`intent`	*Optional*
`leader-offset`	*Optional*
`caption-style`	*Optional*
`caption-offset-h`	*Optional*
`caption-offset-v`	*Optional*

Border style attributes

`width`	*Optional*
`dashes`	*Optional*
`style`	*Optional*

6.4.8 circle

6.4.8.1 Overview

The `circle` element is a child of the `annots` element and corresponds to the Circle annotation. A Circle annotation displays an ellipse on the page.

6.4.8.2 Content model

```
(contents-richtext? & contents? & popup?)
```

6.4.8.3 Attributes

FDF annotation attributes

`page`	*Required*

Common annotation attributes

`color`	*Optional*
`date`	*Optional*

`flags`	*Optional*
`name`	*Optional*
`rect`	*Required*
`title`	*Optional*

Markup annotation attributes

`creationdate`	*Optional*
`opacity`	*Optional*
`subject`	*Optional*

Border style attributes

`width`	*Optional*
`dashes`	*Optional*
`style`	*Optional*

Border effect attributes

`intensity`	*Optional*
`style`	*Optional*

Circle and Square annotation attributes

`interior-color`	*Optional*
`fringe`	*Optional*

6.4.9 square

6.4.9.1 Overview

The `square` element is a child of the `annots` element and corresponds to the Square annotation. A Square annotation displays a rectangle on the page.

6.4.9.2 Content model

```
(contents-richtext? & contents? & popup?)
```

6.4.9.3 Attributes

FDF annotation attributes

`page`	*Required*

Common annotation attributes

`color`	*Optional*
`date`	*Optional*

flags	*Optional*
name	*Optional*
rect	*Required*
title	*Optional*

Markup annotation attributes

creationdate	*Optional*
opacity	*Optional*
subject	*Optional*

Border style attributes

width	*Optional*
dashes	*Optional*
style	*Optional*

Border effect attributes

intensity	*Optional*
style	*Optional*

Circle and Square annotation attributes

interior-color	*Optional*
fringe	*Optional*

6.4.10 caret

6.4.10.1 Overview

The caret element is a child of the annots element and corresponds to the Caret annotation. A Caret annotation is a visual symbol that indicates the presence of text edits.

6.4.10.2 Content model

(contents-richtext? & contents? & defaultappearance? & popup?)

6.4.10.3 Attributes

FDF annotation attributes

page	*Required*

Common annotation attributes

color	*Optional*
date	*Optional*

`flags`	*Optional*
`Name`	*Optional*
`Rect`	*Required*
`title`	*Optional*

Markup annotation attributes

`creationdate`	*Optional*
`opacity`	*Optional*
`subject`	*Optional*

Caret annotation attributes

`fringe`	*Optional*
`symbol`	*Optional*

6.4.11 polygon

6.4.11.1 Overview

The `polygon` element is a child of the `annots` element and corresponds to the Polygon annotation. The Polygon annotation displays a closed polygon on the page.

6.4.11.2 Content model

```
(vertices & contents-richtext? & contents? & popup?)
```

6.4.11.3 Attributes

FDF annotation attributes

`page`	*Required*

Common annotation *attributes*

`color`	*Optional*
`date`	*Optional*
`flags`	*Optional*
`name`	*Optional*
`rect`	*Required*
`title`	*Optional*

Markup annotation attributes

`creationdate`	*Optional*
`opacity`	*Optional*

`subject` *Optional*

Border style attributes

`width` *Optional*

`dashes` *Optional*

`style` *Optional*

Border effect attributes

`intensity` *Optional*

`style` *Optional*

Polygon and Polyline annotation attributes

`interior-color` *Optional*

`intent` *Optional*

6.4.12 polyline

6.4.12.1 Overview

The `polyline` element is a child of the `annots` element and corresponds to the Polyline annotation. The Polyline annotation is similar to the Polygon, but the first and last vertices are not connected. The polyline element has the same properties as polygon plus `LE` attributes.

6.4.12.2 Content model

```
(vertices & contents-richtext? & contents? & popup?)
```

6.4.12.3 Attributes

FDF annotation attributes

`page` *Required*

Common annotation *attributes*

`color` *Optional*

`date` *Optional*

`flags` *Optional*

`name` *Optional*

`rect` *Required*

`title` *Optional*

Markup annotation attributes

`creationdate` *Optional*

`opacity` *Optional*

`subject` *Optional*

Border style attributes

`width` *Optional*

`dashes` *Optional*

`style` *Optional*

Polygon and Polyline annotation attributes

`interior-color` *Optional*

`head` *Optional*

`tail` *Optional*

`intent` *Optional*

6.4.13 stamp

6.4.13.1 Overview

The `stamp` element is a child of the `annots` element and corresponds to the Rubber Stamp annotation. A Rubber Stamp annotation displays text or graphics intended to look as if they were stamped on the page with a rubber stamp.

If present, the `appearance` child element (the **AP** key in the annotation dictionary) shall take precedence over the icon attribute (**Name** key in the rubber stamp annotation dictionary).

6.4.13.2 Content model

```
(contents-richtext? & contents? & appearance? & popup?)
```

6.4.13.3 Attributes

FDF annotation attributes

`page` *Required*

Common annotation *attributes*

`color` *Optional*

`date` *Optional*

`flags` *Optional*

`name` *Optional*

`rect` *Required*

`title` *Optional*

Markup annotation attributes

`creationdate` *Optional*

`opacity`	*Optional*
`subject`	*Optional*

Stamp annotation attributes

`icon`	*Optional*
`rotation`	*Optional*

6.4.14 ink

6.4.14.1 Overview

The `ink` element is a child of the `annots` element and corresponds to the ink annotation. An ink annotation represents a freehand "scribble" composed of one or more disjoint paths.

6.4.14.2 Content model

(inklist & contents-richtext? & contents? & popup?)

6.4.14.3 Attributes

FDF annotation attributes

`page`	*Required*

Common annotation *attributes*

`color`	*Optional*
`date`	*Optional*
`flags`	*Optional*
`name`	*Optional*
`rect`	*Required*
`title`	*Optional*

Markup annotation attributes

`creationdate`	*Optional*
`opacity`	*Optional*
`subject`	*Optional*

Border style attributes

`width`	*Optional*
`dashes`	*Optional*
`style`	*Optional*

6.4.15 freetext

6.4.15.1 Overview

The `freetext` element is a child of the `annots` element and corresponds to the FreeText annotation. A FreeText annotation displays text directly on the page.

6.4.15.2 Content model

(defaultstyle? & contents-richtext? & contents? & defaultappearance)

6.4.15.3 Attributes

FDF annotation attributes

`page`	*Required*

Common annotation *attributes*

`color`	*Optional*
`date`	*Optional*
`flags`	*Optional*
`name`	*Optional*
`rect`	*Required*
`title`	*Optional*

Markup annotation attributes

`creationdate`	*Optional*
`opacity`	*Optional*
`subject`	*Optional*

Border style attributes

`width`	*Optional*
`dashes`	*Optional*
`style`	*Optional*

Freetext annotation attributes

`rotation`	*Optional*
`justification`	*Optional*
`intent`	*Optional*

6.4.16 fileattachment

6.4.16.1 Overview

The `fileattachment` element is a child of the `annots` element and corresponds to a FileAttachment annotation. A FileAttachment annotation contains a reference to a file, which typically will be embedded in the PDF file.

6.4.16.2 Content model

(data & resource? & contents-richtext? & contents?)

6.4.16.3 Attributes

FDF annotation attributes

`page`	*Required*

Common annotation *attributes*

`color`	*Optional*
`date`	*Optional*
`flags`	*Optional*
`name`	*Optional*
`rect`	*Required*
`title`	*Optional*

Markup annotation attributes

`creationdate`	*Optional*
`opacity`	*Optional*
`subject`	*Optional*

Fileattachment annotation attributes

`icon`	*Optional*

Embedded file parameter attributes

`size`	*Optional*
`modification`	*Optional*
`creation`	*Optional*
`checksum`	*Optional*

File specification attributes

`file` *Optional*

Miscellaneous attributes

`mimetype` *Optional*

6.4.17 sound

6.4.17.1 Overview

The `sound` element is a child of the `annots` element and corresponds to the sound annotation. A sound annotation is analogous to a text annotation, except that instead of a text note, it contains sound data.

6.4.17.2 Content model

```
(data & resource? & contents-richtext? & contents?)
```

6.4.17.3 Attributes

FDF annotation attributes

`page` *Required*

Common annotation *attributes*

`color` *Optional*

`date` *Optional*

`flags` *Optional*

`name` *Optional*

`rect` *Required*

`title` *Optional*

Markup annotation attributes

`creationdate` *Optional*

`opacity` *Optional*

`subject` *Optional*

Sound annotation attributes

`icon` *Optional*

`rate` *Required*

`bits` *Optional*

`channels` *Optional*

`encoding` *Optional*

6.4.18 link

6.4.18.1 Overview

The `link` element is a child of the `annots` element and corresponds to the link annotation. A link annotation identifies an area of the document where a link is to be available, and an action to perform or destination to go to should the link be activated.

6.4.18.2 Content model

(contents? & (Dest | OnActivation) & BorderStyleAlt? & popup?)

6.4.18.3 Attributes

FDF annotation attributes

`page`	*Required*

Common annotation *attributes*

`color`	*Optional*
`date`	*Optional*
`flags`	*Optional*
`name`	*Optional*
`rect`	*Required*

Border effect attributes

`style`	*Optional*

Link annotation attributes

`Highlight`	*Optional*
`coords`	*Optional*

6.4.19 redact

6.4.19.1 Overview

The `redact` element is a child of the `annots` element and corresponds to the Redact annotation. A Redact annotation identifies content that is intended to be removed from the document. Redaction is a two-step process in which the user first applies redaction annotations that specify the pieces or regions of content that should be removed and subsequently instructs the PDF processor to apply the redact annotations and remove the content.

6.4.19.2 Content model

(contents-richtext? & contents? & popup? & defaultappearance? & overlayappearance?)

6.4.19.3 Attributes

FDF annotation attributes

`page`	*Required*

Common annotation *attributes*

`color`	*Optional*
`date`	*Optional*
`flags`	*Optional*
`name`	*Optional*
`rect`	*Required*

Redaction annotation attributes

`coords`	*Optional*
`interior-color`	*Optional*
`overlay-text`	*Optional*
`overlay-text-repeat`	*Optional*
`justification`	*Optional*

6.4.20 projection

6.4.20.1 Overview

A projection annotation represents a comment made on a spatial model such as 3D artwork. The appearance of the annotation in a PDF page's two-dimensional coordinate system can be portrayed only as a projection of the comment's appearance in the spatial model. A projection annotation can be created, and its geometry and appearance can be modified, only within the context of an associated runtime environment, such as an activated 3D or geospatial model.

6.4.20.2 Content model

Empty

6.4.20.3 Attributes

FDF annotation attributes

`page`	*Required*

Common annotation *attributes*

`color`	*Optional*
`date`	*Optional*
`flags`	*Optional*
`name`	*Optional*

`rect` *Required*

`title` *Optional*

Markup annotation attributes

`creationdate` *Optional*

`opacity` *Optional*

`subject` *Optional*

6.5 Annotation sub-elements

6.5.1 Action

6.5.1.1 Overview

The `Action` element is a child of the `OnActivation` sub-element of the link element and indicates an action for the interactive PDF processor to perform, such as launching an application or opening a new window. Corresponds to the **A** key in the annotation dictionary.

6.5.1.2 Content model

(URI | Launch | GoTo | GoToR | Named)

6.5.1.3 Attributes

None

6.5.2 appearance

6.5.2.1 Overview

The `appearance` element is a child of the stamp element and corresponds to the **AP** key in the annotation dictionary. The value is a base 64 encoded string.

6.5.2.2 Content model

Base 64 encoded string

6.5.2.3 Attributes

None

6.5.3 BorderStyleAlt

6.5.3.1 Overview

`BorderStyleAlt` is a child of the `link` element and corresponds to the **Border** key in the common annotation dictionary.

6.5.3.2 Content model

Border style encoded in the format specified in the border style attributes

6.5.3.3 Attributes

Border array attributes

`HCornerRadius`	*Required*
`VCornerRadius`	*Required*
`Width`	*Required*
`DashPattern`	*Optional*

6.5.3.4 Details

This format differs from the border style dictionary defined in the **BS** entry in the same table (represented in XDF by style, width, and dashes). The **BS** style of border specification is more recently defined, but array-style borders are still prevalent.

6.5.4 contents

6.5.4.1 Overview

The `contents` element is a child of `caret`, `circle`, `fileattachment`, `freetext`, `highlight`, `ink`, `line`, `link`, `polygon`, `polyline`, `redact`, `sound`, `square`, `squiggly`, `stamp`, `strikeout`, `text`, and `underline`.

This element corresponds to the common annotation key **Contents** in the annotation dictionary.

6.5.4.2 Content model

Text string

6.5.4.3 Attributes

None

6.5.4.4 Details

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 disabled users or for other purposes.

6.5.5 contents-richtext

6.5.5.1 Overview

The `contents-richtext` element is a child of `caret`, `circle`, `fileattachment`, `freetext`, `highlight`, `ink`, `line`, `polygon`, `polyline`, `redact`, `sound`, `square`, `squiggly`, `stamp`, `strikeout`, `text`, and `underline`.

This element corresponds to the **RC** key in the markup annotation dictionary, which provides a rich text string to be displayed in the pop-up window when the annotation is opened.

6.5.5.2 Content model

Text string or rich text string.

See 5.8.3.

6.5.5.3 Attributes

None

6.5.6 data

6.5.6.1 Overview

The `data` element is a child of the `fileattachment` and `sound` elements and contains the encoded file or sound data.

6.5.6.2 Content model

String encoded in the format specified in the mode and encoding attributes

6.5.6.3 Attributes

Miscellaneous attributes

`mode`	*Required*
`encoding`	*Required*

Miscellaneous attributes

`length`	*Required*
`filter`	*Optional*

6.5.6.4 Details

The stream data in the `data` element is output as described in the section titled Stream encoding.

6.5.7 defaultappearance

6.5.7.1 Overview

The `defaultappearance` element is a child of the `caret` and `freetext` elements and corresponds to the **DA** key in the free text annotation dictionary (see 6.5.8 for the `defaultappearance` element that is a child of the `redact` element). The element specifies the default appearance string to be used in formatting the text.

6.5.7.2 Content model

Text string

6.5.7.3 Attributes

None

6.5.8 defaultappearance

6.5.8.1 Overview

The `defaultappearance` element is a child of the `redact` element and corresponds to the **DA** key in the redaction annotation dictionary (see 6.5.7 for the `defaultappearance` element that is a child of the `caret` and `freetext` elements). The value specifies the appearance string to be used in formatting the overlay text when it is drawn after the affected content has been removed. Its value shall be ignored if `overlayappearance` is present.

6.5.8.2 Content model

Text string

6.5.8.3 Attributes

None

6.5.9 defaultstyle

6.5.9.1 Overview

The `defaultstyle` element is a child of the `freetext` element and corresponds to the **DS** key in the free text annotation dictionary. A default style string.

6.5.9.2 Content model

Text string

6.5.9.3 Attributes

None

6.5.10 Dest

6.5.10.1 Overview

The `Dest` element is a child of the `link`, `GoTo`, and `GoToR` elements and corresponds to the **Dest** key in the link annotations dictionary.

6.5.10.2 Content model

(Named | XYZ | Fit | FitH | FitV | FitR | FitB | FitBH | FitBV)

6.5.10.3 Attributes

None

6.5.10.4 Details

The target of the link is specified as a `name`, `string` or `array`.

6.5.11 File

6.5.11.1 Overview

The `File` element is a child of the `GoToR` and `Launch` elements and corresponds to the **F** key in the remote go-to actions and launch dictionaries.

6.5.11.2 Content model

None

6.5.11.3 Attributes

File specification attributes

`OriginalName` *Required*

6.5.12 gesture

6.5.12.1 Overview

The `gesture` element is a child of the `inklist` element and contains the data from the **InkList** array.

6.5.12.2 Content model

Text string

6.5.12.3 Attributes

None

6.5.12.4 Details

The `gesture` element contains a `text string` made up of pairs of comma-separated real numbers separated by a semicolon. The pairs of real numbers represent a horizontal or vertical coordinate. Horizontal and vertical coordinates pairs represent a path. Therefore, the semicolon separated coordinates also occur in pairs.

Here is an example of the gesture element:

```
<gesture>87.712692,451.954437;85.805893,453.225616</gesture>
```

6.5.13 Fit

6.5.13.1 Overview

The `Fit` element is a child of the `Dest` element and corresponds to the **Fit** key in the destination syntax.

6.5.13.2 Content model

None

6.5.13.3 Attributes

Destination syntax attributes

`Page`	*Required*

6.5.13.4 Details

Fit displays the page designated by Page, with its contents magnified just enough to fit the entire page within the window both horizontally and vertically.

6.5.14 FitB

6.5.14.1 Overview

The `FitB` element is a child of the `Dest` element and corresponds to the **FitB** key in the destination syntax.

6.5.14.2 Content model

None

6.5.14.3 Attributes

Destination syntax attributes

`Page`	*Required*

6.5.14.4 Details

`FitB` displays the page designated by `Page`, with its contents magnified just enough to fit its bounding box entirely within the window both horizontally and vertically.

6.5.15 FitBH

6.5.15.1 Overview

The `FitBH` element is a child of the `Dest` element and corresponds to the **FitBH** key in the destination syntax.

6.5.15.2 Content model

None

6.5.15.3 Attributes

Destination syntax attributes

`Page`	*Required*
`Top`	*Required*

6.5.15.4 Details

`FitBH` displays 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.

6.5.16 FitBV

6.5.16.1 Overview

The `FitBV` element is a child of the `Dest` element and corresponds to the **FitBV** key in the destination syntax.

6.5.16.2 Content model

None

6.5.16.3 Attributes

Destination syntax attributes

`Page`	*Required*
`Left`	*Required*

6.5.16.4 Details

`FitBV` displays 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.

6.5.17 FitH

6.5.17.1 Overview

The `FitH` element is a child of the `Dest` element and corresponds to the **FitH** key in the destination syntax.

6.5.17.2 Content model

None

6.5.17.3 Attributes

Destination syntax attributes

`Page`	*Required*
`Top`	*Required*

6.5.17.4 Details

`FitH` displays 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.

6.5.18 FitR

6.5.18.1 Overview

The `FitR` element is a child of the `Dest` element and corresponds to the **FitR** key in the destination syntax.

6.5.18.2 Content model

None

6.5.18.3 Attributes

Destination syntax attributes

`Page`	*Required*
`Left`	*Required*
`Bottom`	*Required*
`Right`	*Required*
`Top`	*Required*

6.5.18.4 Details

`FitR` displays 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.

6.5.19 FitV

6.5.19.1 Overview

The `FitV` element is a child of the `Dest` element and corresponds to the **FitV** key in the destination syntax.

6.5.19.2 Content model

None

6.5.19.3 Attributes

Destination syntax attributes

`Page`	*Required*
`Left`	*Required*

6.5.19.4 Details

`FitV` displays 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 width of the page within the window.

6.5.20 GoTo

6.5.20.1 Overview

The `GoTo` element is a child of the `Action` element and corresponds to the **GoTo** key in the action types dictionary.

6.5.20.2 Content model

Dest

6.5.20.3 Attributes

None

6.5.21 GoToR

6.5.21.1 Overview

The GoToR element is a child of the Action element and corresponds to the **GoToR** key in the action types dictionary.

6.5.21.2 Content model

(File and Dest)

6.5.21.3 Attributes

Remote go-to attributes

`NewWindow`	*Optional*

6.5.22 inklist

6.5.22.1 Overview

The `inklist` element is a child of the `ink` element and corresponds to the **InkList** key in the `Ink` annotation dictionary.

6.5.22.2 Content model

gesture+

6.5.22.3 Attributes

None

6.5.22.4 Details

The inklist element contains a series of gestures, each representing a stroked path. Each gesture 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.

6.5.23 Launch

6.5.23.1 Overview

The `Launch` element is a child of the `Action` element and corresponds to the **Launch** key in the action types dictionary.

6.5.23.2 Content model

File

6.5.23.3 Attributes

Launch attributes

NewWindow	*Optional*

6.5.24 Named

6.5.24.1 Overview

The Named element is a child of the Action element and corresponds to the **Named** key in the action types dictionary.

6.5.24.2 Attributes

Named action attributes

Name	*Required*

6.5.25 Named

6.5.25.1 Overview

The Named element is a child of the Dest element and allows a destination to be referred to indirectly by means of a name object or a byte string.

6.5.25.2 Attributes

Destination syntax attributes

Name	*Required*

6.5.26 OnActivation

6.5.26.1 Overview

The OnActivation element is a child of the link element and corresponds to the A key in the link annotation dictionary.

6.5.26.2 Content model

Action

6.5.26.3 Attributes

None

6.5.27 overlayappearance

6.5.27.1 Overview

The `overlayappearance` element is a child of the `redact` element and corresponds to the **RO** key in the Redaction annotation dictionary. Its value is a form XObject specifying the overlay appearance for this redaction annotation. After this redaction is applied and the affected content has been removed, the overlay appearance should be drawn such that its origin lines up with the lower-left corner of the annotation rectangle. Takes precedence over the `interior-color`, `overlay-text`, `default-appearance`, and `justification` attributes.

6.5.27.2 Content model

Text string

6.5.27.3 Attributes

None

6.5.28 popup

6.5.28.1 Overview

The `popup` element is a child of the `caret`, `circle`, `fileattachment`, `freetext`, `highlight`, `ink`, `line`, `link`, `polygon`, `polyline`, `redact`, `sound`, `square`, `squiggly`, `stamp`, `strikeout`, `text`, and `underline` elements. This element corresponds to the Popup annotation which is described by the **Popup** key in the annotation dictionary. The popup annotation typically is associated with a parent annotation and is used for editing the parent's text.

6.5.28.2 Content model

Empty

6.5.28.3 Attributes

Common annotation attributes

`color`	*Optional*
`date`	*Optional*
`flags`	*Optional*
`name`	*Optional*
`rect`	*Required*
`title`	*Optional*

Popup annotation attributes

`open`	*Optional*

6.5.29 resource

6.5.29.1 Overview

As of PDF 2.0, this entry has been deprecated.

The `resource` element is a child of the `fileattachment` element and corresponds to the **ResFork** key in the Mac OS file information dictionary.

6.5.29.2 Content model

`String encoded in the format specified in the mode and encoding attributes`

6.5.29.3 Attributes

Miscellaneous attributes

`mode`	*Required*
`encoding`	*Required*

Stream attributes

`length`	*Required*
`filter`	*Required*

Mac OS file information *attributes*

`creator`	*Optional*
`subtype`	*Optional*

6.5.29.4 Details

The `resource` element contains the binary contents of the embedded file's resource fork. The data in the resource element is output as described in the section titled "Stream encoding".

6.5.30 URI

6.5.30.1 Overview

The `URI` element is a child of the `Action` element and corresponds to the **URI** key in the action types dictionary.

6.5.30.2 Content model

None

6.5.30.3 Attributes

URI attributes

`Name`	*Required*
`IsMap`	*Optional*

6.5.31 vertices

6.5.31.1 Overview

The `vertices` element is a child of the `polygon` and `polyline` elements and corresponds to the **Vertices** key in the polygon or polyline annotation dictionary.

6.5.31.2 Content model

Text string

6.5.31.3 Attributes

None

6.5.31.4 Details

An array of alternating horizontal and vertical coordinates of each vertex in default user space. The vertices element contains pairs of comma separated real numbers representing a coordinate. Multiple pairs are separated by a semicolon.

6.5.32 XYZ

6.5.32.1 Overview

The `XYZ` element is a child of the `Dest` element and corresponds to the **XYZ** key in the destination syntax.

6.5.32.2 Content model

None

6.5.32.3 Attributes

Destination syntax attributes

`Page`	*Required*
`Left`	*Required*
`Bottom`	*Required*
`Right`	*Required*
`Top`	*Required*

6.6 Annotation attributes

6.6.1 Overview

All annotation attribute values are of XML string type.

6.6.2 FDF annotation attributes

Name	Description
page	*Required*. The page attribute corresponds to the **Page** key in the FDF annotation dictionary. The page attribute represents the ordinal page number on which this annotation should appear, where page 0 is the first page. Elements: caret, circle, fileattachment, freetext, highlight, ink, line, link, polygon, polyline, projection, redact, sound, square, squiggly, stamp, strikeout, text, and underline

6.6.3 Common annotation attributes

Name	Description
color	*Optional*. The color attribute corresponds to the **C** key. The **C** key contains an array of three numbers between 0,0 and 1,0 in the **DeviceRGB** colour space. Each colour is mapped to a value between 0 and 255 then converted to hexadecimal (00 to FF). The three hexadecimal values are concatenated and prefixed with a hash sign: color="#FFFF00" Elements: caret, circle, fileattachment, freetext, highlight, ink, line, link, polygon, polyline, projection, redact, sound, square, squiggly, stamp, strikeout, text, and underline
date	*Optional*. Corresponds to the **M** key. The preferred format is a PDF date string, but PDF processors should be prepared to display a string in any format. Elements: caret, circle, fileattachment, freetext, highlight, ink, line, link, polygon, polyline, projection, redact, sound, square, squiggly, stamp, strikeout, text, and underline
flags	*Optional*. Default: is no flags. Corresponds to the **F** key. A set of flags specifying various characteristics of the field's widget annotation.
	The value is a comma separated list containing zero or more of the following values: — invisible — hidden — print — nozoom — norotate — noview — readonly — locked — togglenoview Example: flags="print,locked" Elements: caret, circle, fileattachment, freetext, highlight, ink, line, link, polygon, polyline, projection, redact, sound, square, squiggly, stamp, strikeout, text, and underline

Name	Description
name	*Optional*. Corresponds to the **NM** key. A string containing the annotation name, a text string uniquely identifying it among all the annotations on its page. Elements: caret, circle, fileattachment, freetext, highlight, ink, line, link, polygon, polyline, projection, redact, sound, square, squiggly, stamp, strikeout, text, and underline
rect	*Required*. Corresponds to the **Rect** key. The annotation rectangle, defining the location of the annotation on the page in default user space units. The value is four comma separated real numbers which may be positive or negative. Elements: caret, circle, fileattachment, freetext, highlight, ink, line, link, polygon, polyline, projection, redact, sound, square, squiggly, stamp, strikeout, text, and underline
title	*Optional*. Corresponds to the **T** key. The text label to be displayed in the title bar of the annotation's popup window when open and active. Elements: caret, circle, fileattachment, freetext, highlight, ink, line, polygon, polyline, projection, sound, square, squiggly, stamp, strikeout, text, and underline

6.6.4 Markup annotation attributes

Name	Description
creationdate	*Optional*. Corresponds to the **CreationDate** entry. The date and time when the annotation was created. Value is in PDF date format. NOTE: The format for creationdate is intentionally different than that of date in 6.6.3. Elements: caret, circle, fileattachment, freetext, highlight, ink, line, polygon, polyline, projection, sound, square, squiggly, stamp, strikeout, text, underline, and projection
opacity	*Optional*. Default is 1,0. Value is a decimal number. Corresponds to the **CA** key. The constant opacity value to be used in painting the annotation. This value applies to all visible elements of the annotation in its closed state (including its background and border), but not to the popup window that appears when the annotation is opened. The specified value is not used if the annotation has an appearance stream; in that case, the appearance stream itself must specify any desired transparency. The implicit blend mode is Normal. Elements: caret, circle, fileattachment, freetext, highlight, ink, line, polygon, polyline, sound, square, squiggly, stamp, strikeout, text, underline, and projection
subject	*Optional*. Corresponds to the **Subj** key. Text representing a short description of the subject being addressed by the annotation. The value is a string. Elements: caret, circle, fileattachment, freetext, highlight, ink, line, polygon, polyline, projection, sound, square, squiggly, stamp, strikeout, text, underline, and projection
intent	*Optional*. A name describing the intent of the markup annotation. Corresponds to the **IT** key in the markup annotation dictionary. Intents allow PDF processors to distinguish between different uses and behaviours 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 PDF processor. Elements: freetext, line, polygon, and polyline

6.6.5 Text markup annotation attributes

Name	Description
`coords`	*Required*. Corresponds to the **QuadPoints** key in the text markup annotation dictionary. Value is one or more groups of 8 comma separated real numbers. Groups are separated by commas. An array of 8 × 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 following order: `x1,y1,x2,y2,x3,y3,x4,y4` These specify the quadrilateral's four vertices in counterclockwise order. The text is oriented with respect to the edge connecting points `(x1,y1)` and `(x2,y2)`. Elements: `highlight, squiggly, strikeout` and `underline`
`inreplyto`	*Required if* `replyType` *is present, otherwise optional*. Corresponds to the **IRT** key in the markup annotation dictionary. A reference to the annotation to which this annotation is in reply. Both annotations must be on the same page of the document. The value is not a dictionary but a text string containing the contents of the name attribute of the annotation being replied to, to allow for a situation where the annotation being replied to is not in the same file. Elements: `text`
`replyType`	*Optional, only meaningful if* `inreplyto` *is present*. Default value is `reply`. A name specifying the relationship (the "reply type") between this annotation and the one specified by `inreplyto`. Corresponds to the **RT** key in the markup annotation dictionary. Values are: — `reply` (default) — `group` Elements: `text`

6.6.6 Text annotation attributes

Name	Description
`icon`	*Optional*. The icon attribute corresponds to the **Name** key in the text annotation dictionary. The name of the icon to be used in displaying the annotation. PDF processors should provide predefined icon appearances for at least the following standard names: — `Comment` — `Check` — `Circle` — `Cross` — `Help` — `Insert` — `Key` — `NewParagraph` — `Note` (default) — `Paragraph` — `RightArrow` — `RightPointer` — `Star` — `UpArrow` — `UpLeftArrow` Additional names may be supported as well. Elements: `text`
`state`	*Optional*. The state to which the annotation should be set. The state attribute corresponds to the State key in the text annotation dictionary. If `statemodel` is set to Marked, the default value is `Unmarked`. If `statemodel` is set to `Review`, the default value is `None`. Values are: — `Marked` — `Unmarked` — `Accepted` — `Rejected` — `Cancelled` — `Completed` — `None` Elements: `text`
`statemodel`	*Required if* `state` *is present, otherwise optional*. The statemodel attribute corresponds to the **StateModel** key in the text annotation dictionary. Values are: — `Marked` — `Review` Elements: `text`

6.6.7 Line annotation attributes

Name	Description
`start`	*Required*. Two comma separated real numbers specify the starting coordinates. Corresponds to the first two numbers in the **L** key in the line annotation dictionary. The **L** key is an array of four numbers specifying the starting and ending coordinates of the line in default user space. Elements: `line`
`end`	*Required*. Two comma separated real numbers specify the starting coordinates. Corresponds to the first two numbers in the **L** key in the line annotation dictionary. The **L** key is an array of four numbers specifying the starting and ending coordinates of the line in default user space. Elements: `line`
`head`	*Optional*. Default: `None`. The line end for the head. Corresponds to first name in the **LE** key in the line annotation dictionary. The **LE** key is an array of two names specifying the line ending styles to be used in drawing the line. Values are: — `None` (default) — `Square` — `Circle` — `Diamond` — `OpenArrow` — `ClosedArrow` — `Butt` — `ROpenArrow` — `RClosedArrow` Elements: `line`
`tail`	*Optional*. Default: `None`. The line end for the tail. Corresponds to second name in the **LE** key in the line annotation dictionary. The **LE** key is an array of two names specifying the line ending styles to be used in drawing the line. Values are: — `None` (default) — `Square` — `Circle` — `Diamond` — `OpenArrow` — `ClosedArrow` — `Butt` — `ROpenArrow` — `RClosedArrow` Elements: `line`