Adobe

Adobe® PDF

XML Forms Data Format Specification

August 2009

Version 3.0

XML Forms Data Format Specification

Version 3.0, August 2009

The content of this guide is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Adobe Systems Incorporated. Adobe Systems Incorporated assumes no responsibility or liability for any errors or inaccuracies that may appear in the informational content contained in this guide.

Any references to company names, company logos and user names in sample material or sample forms included in this documentation and/or software are for demonstration purposes only and are not intended to refer to any actual organization or persons.

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, USA.

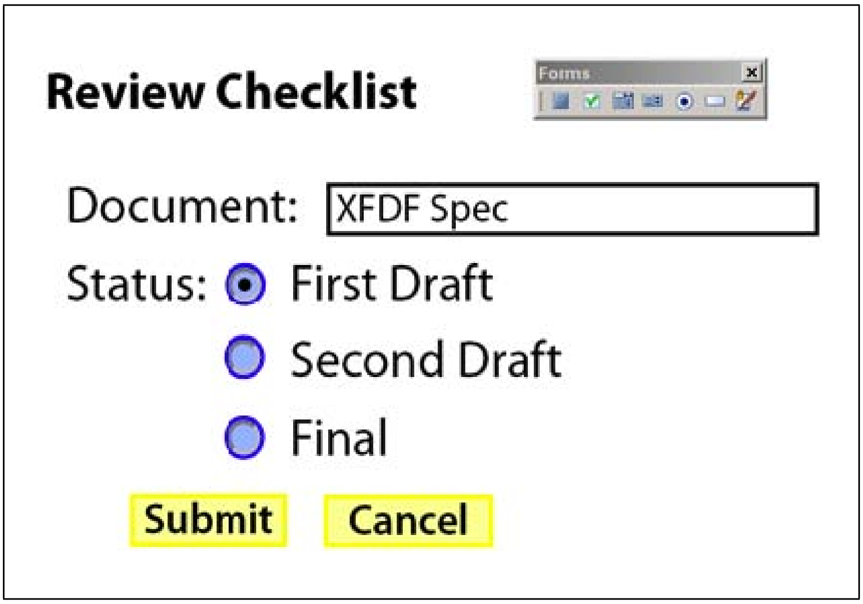
Contents

1 Introduction to XFDF

XFDF (XML Forms Data Format) is a format for representing forms data and annotations in a PDF document. This specification describes XFDF compatible with Adobe Extension Level 3 to ISO 32000-1 (PDF 1.7) and Acrobat 9.0. XFDF is the XML version of Forms Data Format (FDF), a simplified version of PDF for representing forms data and annotations.

Forms Data and Annotations

Form fields in a PDF document include edit boxes, buttons, and radio buttons, as shown in the following PDF document:



Review Checklist

Document: XFDF Spec

Status: ● First Draft

○ Second Draft

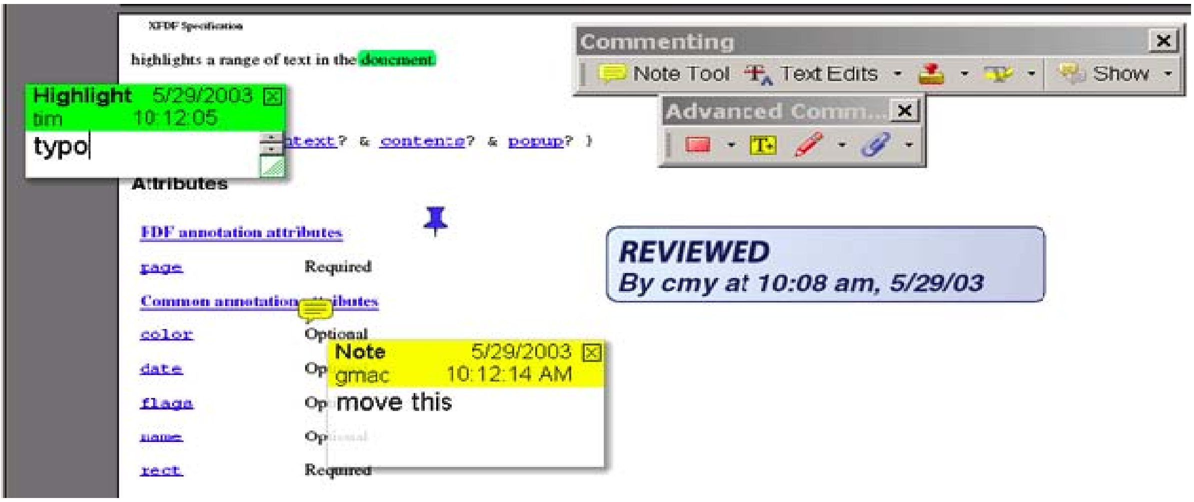
○ Final

The XFDF exported from this PDF document looks like the following:

```
<?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>
```

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

Annotations are attached to a PDF document, and include text notes, highlights, stamps, and file attachments as shown in the following PDF document:



XFDF Specification

highlights a range of text in the document

Highlight 5/29/2003
tim 10:12:05
typo

text? & contents? & popup?)

Attributes

FDF annotation attributes

page	Required

Common annotation attributes

color	Optional
date	Op
flags	Op
name	Optional
rect	Required

Note 5/29/2003
gmac 10:12:14 AM
move this

REVIEWED
By cmy at 10:08 am, 5/29/03

How to Use This Specification

This specification documents the correspondence between XFDF element or attribute and PDF dictionary and key. A short description is provided for each element and attribute; for complete information, look in the *PDF Reference* under the corresponding dictionary and key. There are a few attributes that do not correspond to a PDF dictionary and key.

This specification contains the following major sections:

- Introduction to XFDF
- PDF, FDF and XFDF - a comparison of the three formats
- Writing XFDF - XML implementation details
- Understanding Forms - includes samples
- Understanding Annotations - how to read or write annotations, including samples
- Implementation Notes - notes about the XFDF implementation and XML

The reference sections are:

- XFDF Elements
- Form Field Elements
- Annotation Elements
- Annotation Subelements
- Annotation attributes
- Mapping Tables - From PDF key to XFDF element or attribute.
- List of References

PDF, FDF and XFDF

PDF, FDF, and XFDF are related specifications with PDF 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 the PDF Reference. XFDF is documented in this specification, and is supplemented by information in the PDF Reference.

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 correspond 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 not 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, which has gained wide acceptance and is supported by many existing XML tools. For example, XML tools supporting XSLT can be used to transform an XFDF file to another format. Currently, Adobe does not provide a schema for validation because the specification cannot be realized in standard XML Schema (XSD). In the future, a schema in Relax NG format may be provided.

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 information may be found in the PDF Reference (look for the `CreationDate` key in the markup annotation dictionary).

This is the `creationdate` attribute in XFDF:

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

This is the `CreationDate` entry in a PDF or FDF:

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

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.

This is the `ids` element in XFDF:

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

this is the corresponding `ID` entry in FDF:

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

Next, we will look at a sample form and annotation in both FDF and XFDF format.

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>
```

Sample annotation in FDF and XFDF

As mentioned before, 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="BEF7724317B311718E8675B677EF9B4E"
  />
  <annots>
    <text flags="print,nozoom,norotate" page="0" subject="Note"
      rect="271.850464,690.255371,291.850464,708.255371"
      creationdate="D:20030425095243-07'00'"
      name="apYVRecPEj75sYIwSxME7C" ...
    >
      ...
      <popup .../>
    </text>
  </annots>
</xfdf>
```

Writing XFDF

This section describes XML implementation details specific to XFDF.

Encoding and Namespace

The encoding in the XFDF file must be UTF-8. Each XFDF file begins with the line:

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

The namespace for XFDF is:

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

Thus, an XFDF document begins with these two lines:

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

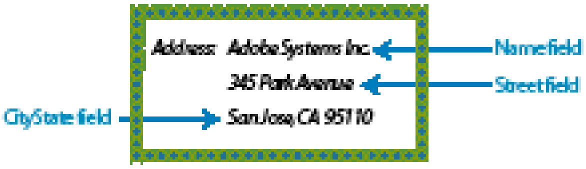
Understanding Forms

An XFDF file with form data contains form field names and values. When importing XFDF into Acrobat, 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.

Simple XFDF form

This simple example shows a PDF document for an address label containing text box form fields named `Name`, `Street` and `CityState`. The PDF file looks like:



The form data is exported to XFDF using the Acrobat `Advanced > Forms > Export Forms Data...` menu item, and selecting XFDF format. 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 value.

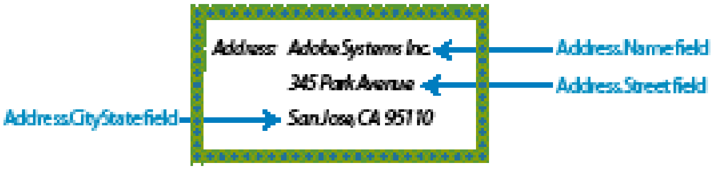
```
<?xml version="1.0" encoding="UTF-8"?>
<xfdf xmlns="http://ns.adobe.com/xfdf/" xml:space="preserve">
  <f href="samples/AddressLabel.pdf"/>
  <ids original="7A0631678ED475F0898815F0A818CFA1"
    modified="BEF7724317B311718E8675B677EF9B4E"/>
  <fields>
    <field name="Name">
      <value>Adobe Systems,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

Hierarchical XFDF form

In Acrobat, hierarchical form fields are 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
```

The PDF file appears the same as in the simple example, but the field names are changed:



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:

```
<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>
<fields>
```

Understanding Annotations

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 annotation, color, 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.

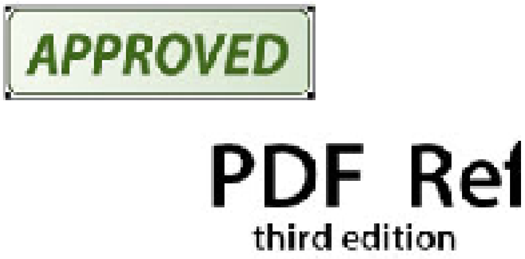
Markup and Popup annotations are represented in XFDF; there are only five annotations that are not represented in XFDF. Each annotation is represented by an element: for example, a Text annotation is represented by the `text` element, and a Polygon annotation is represented by the `polygon` element.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

Supported Annotations	**Unsupported Annotations**
Text	Movie
FreeText	Widget
Line	Screen
Square	PrinterMark
Circle	TrapNet
Polygon	
Polyline	
Highlight	

Supported Annotations	Unsupported Annotations
Underline	
Squiggly	
StrikeOut	
Stamp	
Caret	
Ink	
Popup	
FileAttachment	
Sound	
Link	
Redact	
Projection	

Simple XFDF annotation

In this simple example, a stamp annotation has been applied to a page in a PDF file:



Annotations are exported to XFDF using the Acrobat `Document > Export Comments...` menu item and selecting XFDF as the format.

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.

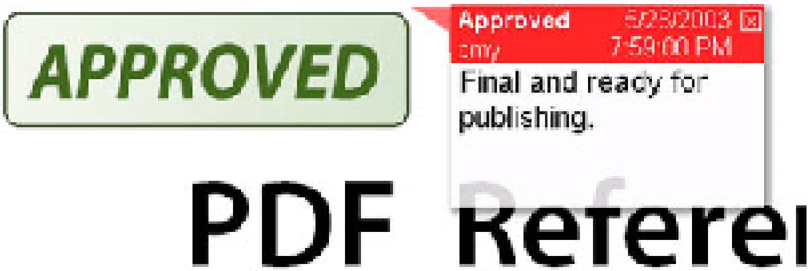
The `stamp` element contains a `popup` element which corresponds to the popup window for adding comments that is attached to the annotation. In this example, the popup window is empty and closed `(open="no").`

```
<?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="BEF7724317B311718E8675B677EF9B4E"/>
 <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="SBApproved"
     color="#FF0000" date="D:20030528192529-07'00'"
     title="cmy">
   <popup flags="print,nozoom,norotate" page="0"
     rect="612.000000,619.065979,792.000000,739.065979"
     open="no"/>
  </stamp>
 </annots>
</xfdf>
```

Annotation with popup text

If the rubber stamp annotation had an open popup note with text, it would look like this in Acrobat:



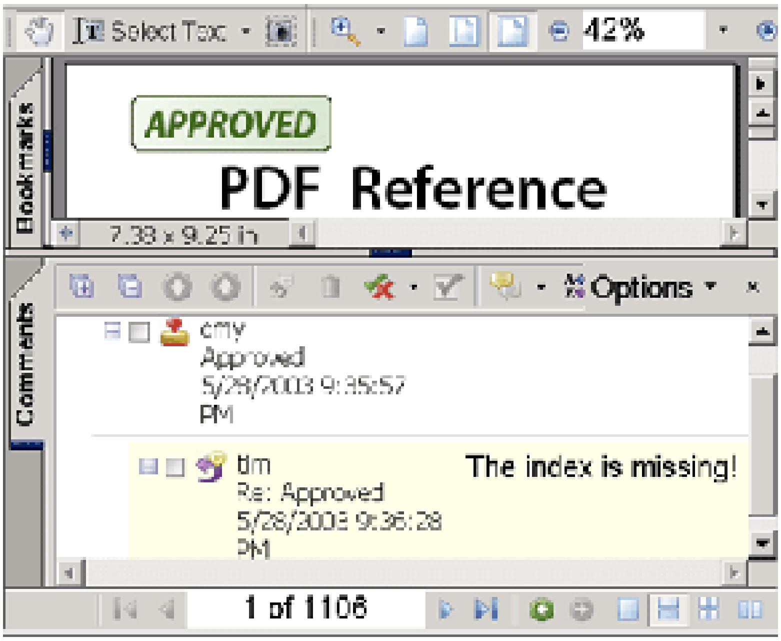
Final and ready for publishing.

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 a rich text strings. For more information on rich text strings see the section below titled Rich text strings . Here is the new stamp element with some attributes removed for readability:

```
<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="Acrobat:6.0.0" xfa:spec="2.0.2">
   <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>
```

Annotation with comment

Annotations can have comments attached to them. In Acrobat, these are displayed in the Comments List window. In this example, the rubber stamp annotation has one comment:



cmy
Approved
5/28/2003 9:35:57 PM

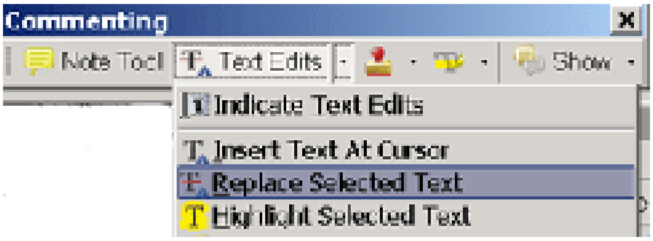
tim
Re: Approved
5/28/2003 9:36:28 PM
The index is missing!

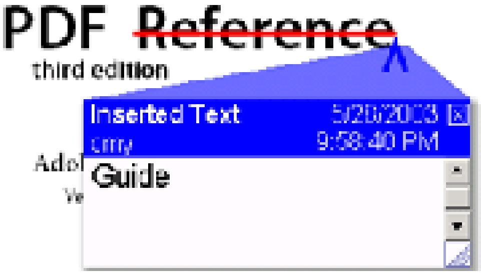
The example below shows the annots element exported from the PDF file (attributes have been removed to improve readability). The comment is contained in the `text` element which is the second child of the annots element and follows the stamp element. The `text` element represents a comment because the 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 the section titled Rich text strings.

```
<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="Acrobat:6.0.0" xfa:spec="2.0.2">
    <p>
     <span style="font-size:10.0pt">The index is missing!</span>
    </p>
   </body>
  </contents-richtext>
  <popup open="no"/>
 </text>
</annots>
```

Strikeout with Caret

The `Commenting > Text Edits > Replace Selected Text` menu item creates two annotations: a Strikeout and Caret annotation.





The `annots` element (with attributes removed for readability) exported from the PDF file contains a `strikeout` followed by a caret element.

```
<annots>
 <strikeout subject="Cross-Out"
    name="8XgvfTdQ6aFx6GdvKcQZGA"
    title="cmy"
coords="264.417999,657.670044,470.810333,657.670044,264.417999,602.998413,47
0.810333,602.998413">
  <popup open="no"/>
 </strikeout>
 <caret flags="print" page="0"
    subject="Inserted Text"
    rect="458.235931,593.156860,483.384735,623.774048"
    name="am_522zM5jow0lHotZX5RC"
    title="cmy" fringe="4.373993,4.373993,4.373993,4.373993">
  <contents-richtext>
   <body xmlns="http://www.w3.org/1999/xhtml"
         xmlns:xfa="http://www.xfa.org/schema/xfa-data/1.0/"
         xfa:APIVersion="Acrobat:6.0.0" xfa:spec="2.0.2">
    <p>
     <span style="font-size:10.0pt">Guide</span>
    </p>
   </body>
  </contents-richtext>
  <popup flags="print,nozoom,norotate" page="0"
     rect="224.334137,520.427856,352.834137,575.427856"
     open="no"/>
 </caret>
</annots>
```

Implementation Notes

This section contains implementation specific notes:

- Importing and exporting XFDF
- Double byte characters
- String encoding conventions
- The border element
- Rich text strings
- The contents and contents-richtext elements in annotations
- The value and value-richtext elements in fields
- Stream encoding
- XML content model syntax

Importing and exporting XFDF

XFDF files can be imported to and exported from Acrobat 6.0 using the following menu items.

- To import and export XFDF annotations, use the Document > Import Comments... and Export Comments... menu items.
- To import and export XFDF form fields, use the Advanced > Forms > Import Forms Data... and Export Forms Data... menu items.

Double byte characters

Although XFDF is encoded in UTF-8, double byte characters are encoded as character references when exported from Acrobat.

For example, the Japanese double byte characters あ , い , and う are exported to XFDF using three character references. Here is an example of double byte characters in a form field:

```
...
<fields>
  <field name="Text1">
     <value>Here are 3 UTF-8 double byte
        characters: &#x3042;&#x3044;&#x3046;
</value>
  </field>
</fields> ...
```

Acrobat can import an XFDF file with double byte UTF-8 characters. The characters do not have to be encoded as character references:

```
...
```

```
<fields>
  <field name="Text1">
     <value>Here are 3 UTF-8 double byte characters:  あ , い , う </value>
  </field>
</fields>
...
```

In Acrobat, set the form field font to one that is able to display Japanese characters (Heisei Kakugo, for example).

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 &#xA; &#xD; and &#x9; 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 the corresponding named entity

Specific ISO 8859-1 Latin-1 Character Conversions

ISO 8859-1 CODE POINT	LATIN-1 CHARACTER NAME	STRING REPRESENTATION
09 (0x9)	HT	&#x9;
10 (0xA)	NL	&#xA;
13 (0xD)	CR	&#xD;
0 - 8 (0x0 - 0x8) 11 - 12 (0xB - 0xC) 14 - 31 (0xE - 0x1F)	NUL - BS VT - NP SO - US	“\000” - “\010” (PDF escape octal code) “\013” - “\014” “\016” - “\037”
34 (0x22)	“	&quot;
38 (0x26)	&	&amp;
60 (0x3c)	<	&lt;
62 (0x3e)	>	&gt;
127 - 159 (0x7f - 0x9f)	DEL - (unassigned)	“\177” - “\237” (PDF escape octal code)

If a schema or DTD for the resulting XML is created, the attributes that are to receive converted PDF string values should be specified as CDATA. This helps to guarantee that any whitespace is preserved. The transformation between the XML representation and the PDF representation is such that converting from PDF to XML then back to XML from PDF reproduces the original binary string.

Encoding Examples

PDF STRING	XML ATTR STRING
Jump in the “Lake” <Jake>	attr=”Jump in the &quot;Lake&quot; &lt;Jake&gt;”
abc123 nothing special here	attr=”abc123 nothing special here”
unusual \200\177\220\237 characters (Here, the \ddd represent an actual single byte character with code octal ddd)	attr=”unusual \200\177\220\237 characters”

Enhancements

Before assuming the encoding is ISO Latin-1, it is permissible to scan the string to determine if it uses UTF-8 encoding. If so, the translation described above can still be used, but the translation should be applied to the UTF-8 characters instead of individual bytes.

The border element

Legacy XFDF files with `freetext` annotations may contain a `border` element:

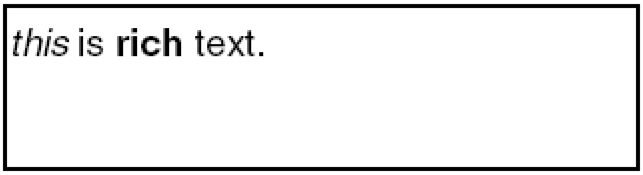
```
<freetext ...>
 ...
 <border width="..."/>
 ...
</freetext>
```

When importing this XFDF annotation to a PDF file, the `border` element is mapped to both the `Border` and BS keys. On a round trip back to XFDF, this will be mapped to the `width` attribute of the `freetext` annotation element.

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 the *PDF Reference*.

For example, the following Text Field form has a value formatted as rich text.

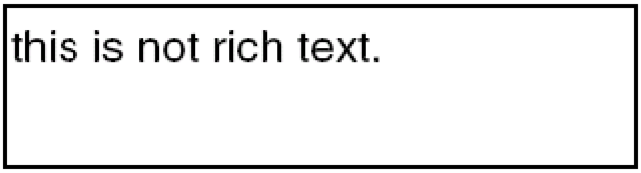


this is **rich** text.

The rich text is mapped to a `value-richtext` element in XFDF:

```
<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="Acrobat:6.0.0" xfa:spec="2.0.2"
    >
      <p>
        <span style="font-size:10.0pt">
        <i>this</i> is <b>rich</b> text.</span>
      </p>
    </body>
  </value-richtext>
</field>
```

Without rich text, the form field would look like:



this is not rich text.

and could be represented by a `value` element in XFDF:

```
<field>
  <value>this is not rich text.</value>
</field>
```

The contents and contents-richtext elements in annotations

Both the `contents` and `contents-richtext` elements in XFDF contain the text to display for an annotation. The `contents` element corresponds to the `Contents` key in the annotation dictionary, and the `contents-richtext` element corresponds to the RC key in the markup annotation dictionary.

An `annotation` element may contain a `contents` element, `contents-richtext` element, or both. The RC key was added in PDF 1.5.

When exporting annotations to XFDF using Acrobat 5, the text of the annotation is written to the `contents` element.

When exporting annotations to XFDF using Acrobat 6, the text of the annotation is written to the `contents-richtext` element. There are two exceptions to this:

1. **PDF** to **XFDF**: If the PDF file contains both `Contents` and RC keys, only the RC key is written to the XFDF file.
2. **XFDF** to **PDF**: The `contents-richtext` element is mapped to the RC key with the following exception: if the `contents-richtext` element contains plain text, it is mapped to the `Contents` key in PDF.

The value and value-richtext elements in fields

The `value` and `value-richtext` elements act similarly to `contents` and `contents-richtext` but are associated with form field values. A `field` element may contain a `value` element, `value-richtext` element, or both.

The `value` and `value-richtext` elements contain the field value. The `value` element corresponds to the V key in the FDF field dictionary, and the `value-richtext` element corresponds to the RV key in the variable text field dictionary. The RV key was added in PDF 1.5.

When exporting form fields to XFDF using Acrobat 5, the text of the form field is written to the `value` element.

When exporting form fields to XFDF using Acrobat 6, the text of the form field is written to the `value-richtext` element. There are two exceptions to this:

1. **PDF** to **XFDF**: If the PDF file contains both V and RV keys, only the RV key is written to the XFDF file.
2. **XFDF** to **PDF**: The `value-richtext` element is mapped to the RV key with the following exception: if the `value-richtext` element contains plain text, it is mapped to the V key in PDF.

Stream encoding

The data of a stream is output to XML with two combinations of the `mode` and `encoding` attributes:

1. `mode="filtered" encoding="ascii"`
2. `mode="raw" encoding="hex"`

Acrobat uses the following tests to determine which method to use when writing out XFDF.

- If the stream is greater than or equal to 4 kilobytes, use method 2.
- If the stream is less than 4 kilobytes and contains only printable ASCII, use method 1; otherwise use method 2.

The 4 kilobyte limit is not a rule; it is the output method used by Acrobat.

Printable ASCII is where each byte of the stream when interpreted as an unsigned integer has a value less than 127 and greater than 32 or is a carriage return or linefeed. The following XML control characters are converted (or filtered) to an entity:

Character	Entity
<	`&lt;`
>	`&gt;`
&	`&amp;`
"	`&quot;`

In method 2, the data is converted to a hexadecimal encoding where each byte is converted to a two character representation of the unsigned integer value, [0-9A-F][0-9A-F]. The high nibble is always first. For example, the ASCII space character is decimal 32 or hex 20. Acrobat adds a linefeed (\n) after each 80 characters of output. The linefeed is not required; however, linefeeds in the data will be handled gracefully.

XML content model syntax

In the Element Reference, 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 the following table.

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.

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:

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

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/>
```

2 XFDF Reference

XFDF Elements

This section describes the top level `xfdf` element and two of its children:

- `xfdf`
- `f`
- `ids`

xfdf

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

Content model

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

Attributes

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

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.

Content model

Empty.

Attributes

`href`	Required. File specification pointing to the source file or target file.

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 file’s trailer dictionary.

Content model

Empty.

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. Value is a hexadecimal number. When assigned by Acrobat, this is an MD5 signature value.
`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. Value is a hexadecimal number. When assigned by Acrobat, this is an MD5 signature value.

Form Field Elements

These elements are used in form fields:

- `fields`
- `field`
- `value`
- `value-richtext`

fields

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.

Content model

`field*`

Attributes

None.

field

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

Content model

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

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.

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>
```

value

The `value` element is a child of the `field` element and contains the field's value, whose format varies depending on the field type. 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.

Content model

Text string.

Attributes

None.

value-richtext

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

Content model

Text string or rich text string. See Rich text strings and the *PDF Reference* for more information.

Attributes

None.

Annotation Elements

This section contains elements used in annotations:

- annots
- text
- highlight
- underline
- strikeout
- squiggly
- line
- circle
- square
- caret
- polygon
- polyline
- stamp
- ink
- freetext
- fileattachment
- sound
- link
- redact
- projection

annots

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.

Content model

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

Attributes

None.

text

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.

Content model

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

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

highlight

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.

Content model

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

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 markup annotation attributes	
coords	Required

underline

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.

Content model

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

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 markup annotation attributes	
coords	Required
intent	Optional

strikeout

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.

Content model

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

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 markup annotation attributes	
coords	Required

squiggly

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.

Content model

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

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 markup annotation attributes	
coords	Required

line

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.

Content model

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

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
Line annotation attributes	
start	Required
end	Required
head	Optional
tail	Optional
interior-color	Optional
leaderLength	Optional
leaderExtend	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

circle

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

Content model

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

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

square

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.

Content model

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

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

caret

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.

Content model

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

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

Polygon and Polyline annotation attributes	
fringe	Optional
symbol	Optional

polygon

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.

Content model

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

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

polyline

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 vertex are not connected. The `polyline` element has the same properties as `polygon` plus LE attributes.

Content model

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

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

stamp

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) takes precedence over the `icon` attribute (`Name` key in the rubber stamp annotation dictionary).

Content model

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

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

ink

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.

Content model

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

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

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.

Content model

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

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

fileattachment

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.

Content model

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

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

sound

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 recorded from the computer's microphone or imported from a file.

Content model

(data & contents-richtext? & contents?)

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

link

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.

Content model

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

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

redact

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 redact annotations that specify the pieces or regions of content that should be removed and subsequently instructs the viewer application to apply the redact annotations and remove the content.

Content model

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

Attributes

Redaction annotation attributes	
coords	Optional
interior-color	Optional
overlay-text	Optional
overlay-text-repeat	Optional
justification	Optional

projection

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.

The `projection` annotation element has the following 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
Markup annotation attributes	
rotation	Optional

If `rect` has zero height or zero width, the `appearance` subelement is omitted from the XFDF file.

Annotation Subelements

These are subelements used in annotations:

- Action
- appearance
- BorderStyleAlt
- contents
- contents-richtext
- data
- defaultappearance child of caret and freetext)
- defaultappearance (child of redact)
- defaultstyle
- Dest
- File
- Fit
- FitB
- FitBH
- FitBV
- FitH
- FitR
- FitV
- gesture
- GoTo
- GoToR
- inklist
- Launch
- Named (child of Action)
- Named (child of Dest)
- OnActivation
- overlayappearance
- popup
- resource
- URI
- vertices
- XYZ

Action

The `Action` element is a child of the `OnActivation` subelement of the `link` element and indicates an action (PDF 1.1) for the viewer application to perform, such as launching an application or opening a new window. Corresponds to the `A` key in the annotation dictionary.

Content model

`( URI | Launch | GoTo | GoToR | Named )`

Attributes

None.

appearance

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.

Content model

Base 64 encoded string.

Attributes

None.

BorderStyleAlt

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

Content model

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

Attributes

Border array attributes

HCornerRadius	Required
VCornerRadius	Required
Width	Required
DashPattern	Optional

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 the older array-style borders are what Acrobat emits even today.

contents

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

Corresponds to the common annotation key Contents in the annotation dictionary.

Content model

Text string.

Attributes

None.

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. See the PDF Reference for more information.

contents-richtext

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

Corresponds to the RC key in the markup annotation dictionary. A rich text string to be displayed in the pop-up window when the annotation is opened.

Content model

Text string or rich text string. See Rich text strings and the *PDF Reference* for more information.

Attributes

None.

data

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

Content model

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

Attributes

Miscellaneous attributes	
mode	Required
encoding	Required
Stream attributes	
length	Required
filter	Required

Details

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

defaultappearance

The `defaultappearance` element is a child of the `caret` and `freetext` elements and corresponds to the DA key in the free text annotation dictionary. Specifies the default appearance string to be used in formatting the text.

Content model

Text string.

Attributes

None.

defaultappearance

The `defaultappearance` element is a child of the `redact` element and corresponds to the DA key in the redaction annotation dictionary. 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. Ignored if `overlayappearance` is present.

Content model

Text string.

Attributes

None.

defaultstyle

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.

Content model

Text string.

Attributes

None.

Dest

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

Content model

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

Attributes

None.

Details

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

File

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.

Content model

None.

Attributes

File specification attributes	
OriginalName	Required

gesture

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

Content model

Text string.

Attributes

None.

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>
```

Fit

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

Content model

None.

Attributes

Destination syntax attributes	
Page	Required

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.

FitB

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

Content model

None.

Attributes

Destination syntax attributes

Page	Required

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.

FitBH

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

Content model

None.

Attributes

Destination syntax attributes	
Page	Required
Top	Required

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.

FitBV

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

Content model

None.

Attributes

Destination syntax attributes	
Page	Required
Left	Required

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.

FitH

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

Content model

None.

Attributes

Destination syntax attributes	
Page	Required
Top	Required

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.

FitR

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

Content model

None.

Attributes

Destination syntax attributes	
Page	Required
Left	Required
Bottom	Required
Right	Required
Top	Required

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.

FitV

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

Content model

None.

Attributes

Destination syntax attributes	
Page	Required
Left	Required

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.

GoTo

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

Content model

Dest

Attributes

None.

GoToR

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

Content model

(File & Dest)

Attributes

Remote go-to attributes	
NewWindow	Optional

inklist

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

Content model

gesture+

Attributes

None.

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.

Launch

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

Content model

File

Attributes

Launch attributes	
NewWindow	Optional

Named

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

Attributes

Named action attributes

Name	Required

Named

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 (PDF 1.1) or a byte string (PDF 1.2)

Attributes

Destination syntax attributes

Name	Required

OnActivation

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

Content model

Action

Attributes

None.

overlayappearance

The overlayappearance element is a child of the redact element and corresponds to the RO key in the Redaction annotation dictionary. 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.

Content model

Text string.

Attributes

None.

popup

The popup element is a child of the caret, circle, fileattachment, freetext, highlight, ink, line, link, polygon , polyline, sound, square, squiggly, stamp , strikeout, text, and underline elements. Corresponds 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.

Content model

Empty.

Attributes

Common annotation attributes	
color	Optional
date	Optional
flags	Optional
name	Optional
rect	Required
title	Optional
Popup annotation attributes	
open	Optional

resource

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

Content model

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

Attributes

Miscellaneous attributes	
mode	Required
encoding	Required
Stream attributes	
length	Required
filter	Required
Mac OS file information attributes	

creator	Optional
subtype	Optional

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.

URI

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

Content model

None.

Attributes

URI attributes

Name	Required
IsMap	Optional

vertices

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.

Content model

Text string.

Attributes

None.

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.

XYZ

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

Content model

None.

Attributes

Destination syntax attributes	
Page	Required
Left	Optional
Top	Optional
Zoom	Optional

Annotation attributes

Attributes are grouped by PDF dictionary that defines the corresponding key.

- FDF annotation attributes
- Common annotation attributes
- Markup annotation attributes
- Text markup annotation attributes
- Text annotation attributes
- Line annotation attributes
- Circle and Square annotation attributes
- Polygon and Polyline annotation attributes
- Freetext annotation attributes
- Stamp annotation attributes
- Fileattachment annotation attributes
- Sound annotation attributes
- Popup annotation attributes
- Link annotation attributes
- Redaction annotation attributes
- Border effect attributes
- Border style attributes
- Border array attributes
- Embedded file parameter attributes
- Stream attributes
- File specification attributes
- Destination syntax attributes
- Remote go-to attributes
- Launch attributes
- Named action attributes
- URI attributes
- Mac OS file information attributes
- Miscellaneous attributes

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, sound, square, squiggly, stamp , strikeout, text, and underline.

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 color space. In XFDF, each color 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, sound, square, squiggly, stamp , strikeout, text, and underline.
date	Optional. Corresponds to the M Key. The preferred format is a PDF date string, but viewer applications should be prepared to display a string in any format. Elements: caret, circle, fileattachment, freetext, highlight, ink, line, link, polygon , polyline, 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. Value is a comma separated list containing the values: ● invisible ● hidden ● print ● nozoom ● norotate ● noview ● readonly ● locked ● togglenoview Example: `flags="print,locked"`

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.
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.
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.

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. Elements: caret circle fileattachment freetext highlight ink line, polygon, polyline, sound, square, squiggly, stamp, strikeout, text, underline, and projection.
opacity	Optional. Default is 1.0. Value is 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. Value is a string. Elements: caret circle fileattachment freetext highlight ink line, polygon, polyline, sound, square, squiggly, stamp, strikeout, text, underline, and projection.

Name	Description
`intent`	Optional. A name describing the intent of the markup annotation. Corresponds to the `IT` key in the markup annotation dictionary. 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. In XFDF 2.0, free text, line, and polygon and polyline annotations have defined intents, whose values are enumerated in the corresponding tables. Elements: `freetext`, `line`, `polygon`, and `polyline`.

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 x 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 `x1,y1,x2,y2,x3,y3,x4,y4` specifying 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. In an XFDF file, 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 XFDF file. Elements: `text`

Name	Description
`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`

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. Viewer applications 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`

Name	Description
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

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 ending coordinates. Corresponds to the second 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.

Name	Description
`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 for head and tail are: ● `None` (Default) ● `Square` ● `Circle` ● `Diamond` ● `OpenArrow` ● `ClosedArrow` ● `Butt` ● `ROpenArrow` ● `RClosedArrow`
`interior-color`	Optional. Corresponds to the `IC` key in the line annotation dictionary and specifies the interior color with which to fill the annotation's line endings. If this entry is absent, the interiors of the line endings are left transparent. The `IC` key contains an array of three numbers between 0.0 and 1.0 in the deviceRGB color space. In XFDF, each color 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. For example: `interior-color="#FFFF00"` Elements: `line`
`leaderLength`	Required if `LeaderExtend` is present; otherwise optional. Default: 0 (no leader lines). Corresponds to the `LL` key in the line annotation dictionary and specifies the length of *leader lines* in default user space that extend from each endpoint of the line perpendicular to the line itself. 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. Elements: `line`
`leaderExtend`	Optional. Default: 0 (no leader line extensions). Value is a non-negative number. Corresponds to the `LLE` key in the line annotation dictionary and specifies the length of *leader line extensions* that extend from the line proper 180 degrees from the leader. Elements: `line`

Name	Description
`caption`	Optional. A flag specifying whether or not the text specified by the contents or contents-richtext entries should be replicated as a caption in the appearance of the line. Corresponds to the `Cap` key in the line annotation dictionary. The text should be rendered in a manner appropriate to the content, taking into account factors such as writing direction. Values: - `yes` - `no` (default) Elements: `line`
`intent`	Optional. A name describing the intent of the line annotation. Corresponds to the `IT` key in the line annotations dictionary. Values: - `LineArrow` - `LineDimension` Elements: `line`
`leader-offset`	Optional. A non-negative number representing the length of the leader line offset. Corresponds to the `LLO` key in the line annotations dictionary. The *leader line offset* is the amount of empty space between the endpoints of the annotation and where the *leader lines* begin. Default: 0 (no leader line offset). Elements: `line`
`caption-style`	Optional. Meaningful only if `caption` is `yes`. A name describing the annotation's caption style. Corresponds to the `CP` key in the line annotation dictionary. Values (PDF 1.7): - `Inline` (default) - `Top` Elements: `line`
`caption-offset-h`	Optional. Default value: 0 (no offset). Meaningful only if `caption` is `yes`. A number specifying the horizontal offset of the caption text from its normal positioning. Corresponds to the first entry in the CO key array in the line annotation dictionary. The horizontal offset is measured 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. Elements: `line`.
`caption-offset-v`	Optional. Default value: 0 (no offset). Meaningful only if `caption` is `yes`. A number specifying the vertical offset of the caption text from its normal positioning. Corresponds to the second entry in the `CO` key array in the line annotation dictionary. The vertical offset is measured perpendicular to the the annotation line, with a positive value indicating a shift up and a negative value indicating a shift down. Elements: `line`.

Circle and Square annotation attributes

Name	**Description**
`interior-color`	Optional. Default is empty string or transparent. Corresponds to the `IC` key in the square or circle annotation dictionary and specifies the interior color with which to fill the annotation's rectangle or ellipse. If this entry is absent the interior of the annotation is left transparent. The IC key contains an array of three numbers between 0.0 and 1.0 in the deviceRGB color space. In XFDF, each color 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. For example: `interior-color="#FFFF00"` Elements: `circle`, `square`.
`fringe`	Optional. The fringe attribute is a rectangle that corresponds to the `RD` key in the circle or square annotation dictionary and is a set of four values describing the numerical differences between two rectangles: the `Rect` entry of the annotation and the actual boundaries of the underlying object. Value is the rectangle is defined by four comma separated real numbers. Elements: `circle`, `square`.

Caret annotation attributes

<table>
<tr><th>Name</th><th colspan="2">Description</th></tr>
<tr><td rowspan="5"><code>symbol</code></td><td colspan="2">Optional. The <code>symbol</code> attribute corresponds to <code>Sy</code> key in the caret annotation dictionary. Value is a name specifying a symbol to be associated with the caret:</td></tr>
<tr><td>XFDF</td><td>PDF</td></tr>
<tr><td><code>none</code> (default)</td><td><code>None</code></td></tr>
<tr><td><code>paragraph</code></td><td><code>P</code></td></tr>
<tr><td colspan="2">Elements: <code>caret</code></td></tr>
<tr><td><code>fringe</code></td><td colspan="2">Optional. The <code>fringe</code> attribute is a rectangle that corresponds to the RD key in the caret annotation dictionary and is a set of 4 values describing the numerical differences between two rectangles: the <code>Rect</code> entry of the annotation and the actual boundaries of the underlying object.
Value is the rectangle is defined by four comma-separated real numbers.
Elements: <code>caret</code></td></tr>
</table>

Polygon and Polyline annotation attributes

Name	Description
`interior-color`	Optional. Default is empty string or transparent. Corresponds to the `IC` key in the polygon or polyline annotation dictionary and specifies the interior color with which to fill the annotation's rectangle or ellipse. If this entry is absent the interior of the annotation is left transparent. The `IC` key contains an array of three numbers between 0.0 and 1.0 in the deviceRGB color space. In XFDF, each color 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. For example: `interior-color="#FFFF00"` Elements: `polygon`, `polyline`
`head`	Optional. Meaningful only for polyline annotations. The line end for the head. Corresponds to first name in the LE key in the polygon and polyline 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: `polyline`
`tail`	Optional. Meaningful only for polyline annotations. 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: `polyline`

Name	Description
`intent`	Optional. A name describing the intent of the polygon or polyline annotation. Corresponds to the IT key in the polygon and polyine annotation dictionary. Values: ● `PolygonCloud` ● `polygon-dimension` ● `polyline-dimension` Elements: `polygon`, `polyline`

Freetext annotation attributes

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td><code>justification</code>
(Optional)</td><td>The <code>justification</code> attribute corresponds to the <code>Q</code> key i the free text annotation dictionary. A code specifying the form of quadding (justification) to be used in displaying the annotation's text:
<table>
<tr><th>XFDF</th><th>PDF</th></tr>
<tr><td><code>left</code> (default)</td><td>0</td></tr>
<tr><td><code>centered</code></td><td>1</td></tr>
<tr><td><code>right</code></td><td>2</td></tr>
</table>
Elements: <code>freetext</code></td></tr>
<tr><td><code>rotation</code></td><td>Optional. Value is an integer. Corresponds to the <code>Rotate</code> key. An integer representing the clockwise rotation in degrees.
Elements: <code>freetext</code></td></tr>
<tr><td><code>intent</code></td><td>Optional. Value is a name describing the intent of the freetext annotation. Corresponds to the IT key in the freetext annotations dictionary.
Values:
● <code>FreeTextCallout</code>
● <code>FreeTextTypeWriter</code>
Elements: <code>freetext</code></td></tr>
</table>

Stamp annotation attributes

Name	**Description**
`icon`	Optional. Default: `Draft`. Corresponds to the Name key in the rubber stamp annotation dictionary. The name of an icon to be used in displaying the annotation. These are the stamp names created by Acrobat 6.0: ● `SBRejected` ● `SHAccepted` ● `SHInitialHere` ● `SHSignHere` ● `SHWitness` ● `SBApproved` ● `SBCompleted` ● `SBConfidential` ● `SBDraft` ● `SBFinal` ● `SBForComment` ● `SBForPublicRelease` ● `SBInformationOnly` ● `SBNotApproved` ● `SBNotForPublicRelease` ● `SBPreliminaryResults` ● `SBVoid` These are the stamp names created by Acrobat 5.0: ● `Approved` ● `AsIs` ● `Confidential` ● `Departmental` ● `Draft` (default) ● `Experimental` ● `Expired` ● `Final` ● `ForComment` ● `ForPublicRelease` ● `NotApproved` ● `NotForPublicRelease` ● `Sold` ● `TopSecret` Additional names may be supported as well. Elements: `stamp`

Name	Description
`rotation`	Optional. Value is an integer. Corresponds to the `Rotate` key. An integer representing the clockwise rotation in degrees. Elements: `stamp`

Fileattachment annotation attributes

Name	Description
`icon`	Optional. The `icon` attribute corresponds to the `Name` key in the file attachment annotation dictionary. The name of an icon to be used in displaying the annotation. Viewer applications should provide predefined icon appearances for at least the predefined values. Additional names may be supported as well. Value may be a predefined value or a string. The predefined values are: ● `Graph` ● `Paperclip` ● `PushPin` (default) ● `Tag` ● Elements: `fileattachment`

Sound annotation attributes

Name	Description
`icon`	Optional. The `icon` attribute corresponds to the `Name` key in the sound annotation dictionary and is 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; additional names may be supported as well. Values are: ● `Speaker` (default) ● `Mic` ● `Ear` Elements: `sound`
`bits`	Optional. Default: 8. Corresponds to the `B` key for a sound object and is an integer describing the number of bits per sample value per channel.
`channels`	Optional. Default: 1. Corresponds to the `C` key for a sound object and is an integer describing the number of sound channels.

`encoding`	Optional. Corresponds to the E key for a sound object and is the encoding format for the sample data. Values are: ● `raw` (default) ● `signed` ● `mulaw` ● `alaw`
`rate`	Required. Corresponds to the R key for a sound object and is a real number describing the sampling rate, in samples per second.

Popup annotation attributes

Name	**Description**
`open`	Optional. A flag specifying whether the annotation should initially be displayed open. Corresponds to the `Open` key in the pop-up annotation dictionary. Values: ● `yes` ● `no` (default) Elements: `popup`.

Link annotation attributes

Name	**Description**
`Highlight`	Optional. Corresponds to the `H` key in the link annotation dictionary. Describes the annotation's highlighting mode, the visual effect to be used when the mouse button is pressed or held down inside its active area. Values: ● `None` ● `Invert` (default) ● `Outline` ● `Push` Elements: `link`

coords	Optional. Corresponds to the QuadPoints key in the link annotation dictionary. Value is one or more groups of 8 comma separated real numbers. Groups are separated by commas. An array of 8 x 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 x1,y1,x2,y2,x3,y3,x4,y4 specifying 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: link

Redaction annotation attributes

Name	Description
coords	Optional. Corresponds to the QuadPoints key in the redaction annotation dictionary. Value is an array of 8 x n numbers specifying the coordinates of n quadrilaterals in default user space. If present, these quadrilaterals denote the content region that is intended to be removed. If this entry is not present, the Rect entry denotes the content region that is intended to be removed. Elements: redact
interior-color	Optional. Corresponds to the IC key in the redaction annotation dictionary. Value is an array of three numbers in the range 0.0 to 1.0 specifying the components, in the DeviceRGB color space, of the interior color with which to fill the redacted region after the affected content has been removed. If this entry is absent, the interior of the redaction region is left transparent. Ignored if the overlayappearance entry is present. .Elements: redact
overlay-text	Optional. Corresponds to the OverlayText key in the redaction annotation dictionary. Value is a text string specifying the overlay text that should be drawn over the redacted region after the affected content has been removed. Ignored if overlayappearance is present. Elements: redact
overlay-text-repeat	Optional. Corresponds to the Repeat key in the redaction annotation dictionary. If true, then the text specified by overlay-text should be repeated to fill the redacted region after the affected content has been removed. Ignored if overlayappearance is present. Default value: false. Elements: redact

Name	Description
justification	Optional. Corresponds to the Q key in the redaction annotation dictonary. Ignored if overlayappearance is present. Values: - 0 Left-justified (default) - 1 Centered - 2 Right-justified Elements: redact

Border effect attributes

Name	Description
intensity	Optional. Default: 0 (meaning no effect). Corresponds to the I key in the border effect dictionary. A number describing the intensity of the effect. It is only considered valid when border effect style is set to cloudy. A higher value indicates more puffs in the cloud. Elements: circle, polygon, polyline, square
style	Optional. Default: solid. These values are appended to the list of style attribute values listed in Border style attributes. Values are: - solid - cloudy

Border style attributes

These attributes correspond to the BS key in the border style dictionary.

Name	Description
width	Optional. Value is a decimal number. Default is 1. Corresponds to the W key in the border style dictionary and specifies the border width in points. If this value is 0, no border is drawn. Elements: circle, freetext, ink, line, polygon, polyline, square, text

<table>
<tr><td><code>dashes</code></td><td colspan="2">Optional. Default is 3. Corresponds to the D key in the border style dictionary. A comma separated list of numbers defining a pattern of dashes and gaps to be used in drawing a dashed border. The dash phase is not specified and is assumed to be 0. For example, a dashes attribute with value "3,2" specifies a border drawn with 3-point dashes alternating with 2-point gaps.
Values are: 1 or more numbers separated by a comma. For example:
● 3
● 3,5
● 4,3,2,3
Elements: circle, freetext, ink, line, polygon, polyline, square, text</td></tr>
<tr><td rowspan="8"><code>style</code></td><td colspan="2">Optional. The <code>style</code> attribute corresponds to the S key in the border style dictionary, which specifies the border style. Values are:</td></tr>
<tr><td>XFDF</td><td>PDF Border Style Dictionary</td></tr>
<tr><td><code>solid</code> (default)</td><td><code>S</code></td></tr>
<tr><td><code>dash</code></td><td><code>D</code></td></tr>
<tr><td><code>bevelled</code></td><td><code>B</code></td></tr>
<tr><td><code>inset</code></td><td><code>I</code></td></tr>
<tr><td><code>underline</code></td><td><code>U</code></td></tr>
<tr><td colspan="2">Elements: circle freetext ink line, polygon, polyline, square, text</td></tr>
</table>

Border array attributes

Name	**Description**
`HCornerRadius`	Required. Corresponds to array index 0 in the `Border` key in the common annotation dictionary. The `HCornerRadius` is a number specifying the horizontal corner radius of the rectangular border. Elements: BorderStyleAlt
`VCornerRadius`	Required. Corresponds to array index 1 in the `Border` key in the common annotation dictionary. The `VCornerRadius` is a number specifying the vertical corner radius of the rectangular border. Elements: BorderStyleAlt
`Width`	Required. Corresponds to array index 2 in the `Border` key in the common annotation dictionary. The `Width` is a number specifying the width of the border; if the `Width` is 0, no border is drawn. Elements: BorderStyleAlt

`DashPattern`	Optional. Corresponds to the optional dash array (array index 3) of the `Border` key in the common annotation dictionary. The `DashPattern` is a comma-separated list of numbers specifying the pattern of dashes and gaps of the border. Elements: `BorderStyleAlt`

Embedded file parameter attributes

The following attributes are from the embedded file parameter dictionary.

Name	Description
`checksum`	Optional. Corresponds to the `CheckSum` key in the embedded file parameter dictionary. The checksum attribute is 16-byte string that is the checksum of the bytes of the uncompressed embedded file. The checksum is calculated by applying the standard MD5 message-digest algorithm to the bytes of the embedded file stream. Elements: `fileattachment`
`creation`	Optional. Value is a PDF date. The creation attribute corresponds to the `CreationDate` key in the embedded file parameter dictionary and is the date and time when the embedded file was created. Elements: `fileattachment`
`modification`	Optional. Value is in PDF date format. The modification attribute corresponds to the `ModDate` key in the embedded file parameter dictionary and is the date and time when the embedded file was last modified. Elements: `fileattachment`
`size`	Optional. The size attribute is an integer corresponding to the `Size` key in the embedded file parameter dictionary and is the size of the embedded file, in bytes. Elements: `fileattachment`

Stream attributes

Name	Description
`length`	Required. Corresponds to the Length key in the stream dictionary. Value is an integer describing the number of bytes in the stream. (There may be an additional EOL marker, preceding endstream, that is not included in the count and is not logically part of the stream data.) Elements: `data`, `resource`

`filter`	Required. Corresponds to the Filter key in the stream dictionary. The name of a filter to be applied in processing the stream data, or comma separated list of such names. Multiple filters should be specified in the order in which they are to be applied. Data is decrypted or uncompressed when the user selects `Save Embedded File to Disk...` from right click menu of the file attachment. Value is single filter name or list of names separated by commas. The filter name is a predefined value or user defined value. The predefined values are: ● `ASCIIHexDecode` ● `ASCII85Decode` ● `LZWDecode` ● `FlateDecode` ● `RunLengthDecode` ● `CCITTFaxDecode` ● `JBIG2Decode` ● `DCTDecode` ● `JPXDecode` ● `Crypt` Elements: `data`, `resource`

File specification attributes

Name	**Description**
`file`	The file attribute corresponds to the F key in the file specification dictionary. Elements: `fileattachment`
`OriginalName`	Required. The `OriginalName` attribute corresponds to the F string in the remote go-to action and launch action dictionaries. Elements: `File`

Destination syntax attributes

Name	**Destination**
`Name`	Required. The `Name` attribute specifies a named destination in the destination syntax allowing a destination to be referred to indirectly by means of a name object (PDF 1.1) or a byte string (PDF 1.2) Elements: `Named`

Name	**Destination**
Page	Corresponds to the page object in the destination syntax. Elements: Fit, FitB, FitBH, FitBV, FitH, FitR, FitV, XYZ
Left	Corresponds to the left object in the destination syntax. Elements: FitBV, FitR, FitV, XYZ
Bottom	Corresponds to the bottom object in the destination syntax. Elements: FitR
Right	Corresponds to the right object in the destination syntax. Elements: FitR
Top	Corresponds to the top object in the destination syntax. Elements: FitBH, FitH, FitR, XYZ
Zoom	Corresponds to the zoom object in the destination syntax. Elements: XYZ

Remote go-to attributes

Name	**Description**
NewWindow	Optional. Corresponds to the NewWindow key in the remote go-to action dictionary. Value is 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. Elements: GoToR

Launch attributes

Name	**Description**
NewWindow	Optional. Corresponds to the NewWindow key in the launch action dictionary. Value is 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. Elements: Launch

Named action attributes

Name	Description
Name	The Name attribute corresponds to the N key in the named actions dictionary. Values: ● NextPage ● PrevPage ● FirstPage ● LastPage Elements: Named

URI attributes

Name	Description
Name	The Name attribute corresponds to the URI key in the URI action dictionary. Value is a string containing the uniform resource identifier to resolve, encoded in 7-bit ASCII. Elements: URI
IsMap	The IsMap attribute corresponds to the IsMap key in the action dictionary. Value is a flag specifying whether to track the mouse position when the URI is resolved. Default value: **false**. Elements: URI

Mac OS file information attributes

Name	Description
creator	Optional. Corresponds to the Creator key in the Mac OS file information dictionary. Value is a string containing the embedded file's creator signature. Elements: resource
subtype	Optional. Corresponds to the Subtype key in the Mac OS file information dictionary. Value is a string containing the embedded file's file type. Elements: resource

Miscellaneous attributes

These attributes do not correspond to a PDF key.

Name	Description
mimetype	Optional. Value is the subtype of the embedded file. The value of this entry must be a first-class name, as defined in the PDF Reference. Names without a registered prefix must conform to the MIME media type names defined in Internet RFC 2046, Multipurpose Internet Mail Extensions (MIME), Part Two: Media Types, with the provision that characters not allowed in names must use the 2-character hexadecimal code format described as Name Objects in the PDF Reference. Elements: fileattachment
mode	Required. Values are: ● filtered ● raw Elements: data, resource
encoding	Required. The encoding format of the element content. Values are: ● ascii ● hex Elements: data, resource

Mapping Tables

The mapping tables show the PDF key and XFDF element or attribute and vice versa.

PDF to XFDF

This table shows the mapping between PDF key and XFDF element or attribute. The E/A column indicates whether the Key corresponds to an XFDF element or attribute. XFDF data, encoding, and mode have no corresponding PDF key.

PDF key	Key value	Dictionary	XFDF	E/A
			data	E
			encoding	A
			mode	A
A		Link annotation	Action	E
A		Annotation	OnActivation	E
AP		Annotation	appearance	E
Annots		FDF	annots	E
B		Sound object	bits	A
Border		Annotation	DashPattern	A
Border		Annotation	HCornerRadius	A
Border		Annotation	VCornerRadius	A
Border		Annotation	Width	A
Border		Link annotation	BorderStyleAlt	E
bottom		Destination syntax annotation	Bottom	A
C		Annotation	color	A
C		Sound object	channels	A
CA		Markup annotation	opacity	A
Cap		Line annotation	caption	A
CheckSum		Embedded file parameter	checksum	A
CO		Line annotation	caption-offset-h	A
CO		Line annotation	caption-offset-v	A
Contents		Annotation	contents	E
CP		Line annotation	caption-style	A
CreationDate		Embedded file parameter	creation	A
CreationDate		Markup annotation	creationdate	A
Creator		Mac OS file information	creator	A
D		Border style	dashes	A
DA		Free text or caret annotation	defaultappearance	E
DA		Redaction annotation	defaultappearance	E
Dest		Link annotation	Dest	E

DS		Free text annotation	defaultstyle	E
E		Sound object	encoding	A
F		Annotation	flags	A
F		FDF	f	E
F		FDF	href	A
F		File specification	file	A
F		File specification	OriginalName	A
F		Remote go-to or launch annotation	File	A
Fields		FDF	field	
Fields		FDF	fields	E
Filter		Stream	filter	A
Fit		Destination syntax	Fit	E
FitB		Destination syntax	FitB	E
FitBH		Destination syntax	FitBH	E
FitBV		Destination syntax	FitBV	E
FitH		Destination syntax	FitH	E
FitR		Destination syntax	FitR	E
FitV		Destination syntax	FitV	E
GoTo		Action type	GoTo	E
GoToR		Action type	GoToR	E
H		Link annotation	Highlight	A
I		Border effect	intensity	A
IC		Redaction annotation	interior-color	A
IC		Square or circle annotation	interior-color	A
ID		FDF	ids	E
ID		FDF	modified	A
ID		FDF	original	A
InkList		Ink annotation	gesture	E
InkList		Ink annotation	inklist	E
IRT		Markup annotation	inreplyto	A
IsMap		Action dictionary annotation	IsMap	A
IT		Markup annotation	intent	A
L		Line annotation	end	A
L		Line annotation	start	A
Launch		Action type	Launch	E
LE		Line annotation	head	A
LE		Line annotation	tail	A

left		Destination syntax annotation	Left	A
Length		Stream	length	A
LL		Line annotation	leaderLength	A
LLE		Line annotation	leaderExtend	A
LLO		Line annotation	leader-offset	A
M		Annotation	date	A
ModDate		Embedded file parameter	modification	A
N		Named action	Name	A
NM		Annotation	name	A
Name		File attachment annotation	icon	A
Name		Rubber stamp annotation	icon	A
Name		Sound annotation	icon	A
Name		Text annotation	icon	A
Named		Action type	Named	E
NewWindow		Remote go-to action annotation	NewWindow	A
NewWindow		Launch parameter annotation	NewWindow	A
Open		Pop-up annotation	open	A
Page		FDF file	page	A
Q		Free text annotation	justification	A
Q		Redaction annotation	justification	A
QuadPoints		Text markup annotation	coords	A
QuadPoints		Link annotation	coords	A
QuadPoints		Redaction annotation	coords	A
R		Sound object	rate	A
RC		Markup annotation	contents-richtext	E
RD		Caret, square or circle annotation	fringe	A
RO		Redaction annotation	overlayappearance	E
RT		Markup annotation	replyType	A
RV		Fields containing variable text	value-richtext	E
Rect		Annotation	rect	A
Repeat		Redaction annotation	overlay-text-repeat	A
ResFork		Mac OS file information	resource	E
right		Destination syntax	Right	A
Root		FDF	xfdf	E
Rotate		Freetext and stamp annotations	rotation	A
S		Border style	style	A
Size		Embedded file parameter	size	A

State		Text annotation	state	A
StateModel		Text annotation	statemodel	A
Subj		Markup annotation	subject	A
Subtype	Caret	Annotation	caret	E
Subtype	Circle	Annotation	circle	E
Subtype	FileAttachment	Annotation	fileattachment	E
Subtype	FreeText	Annotation	freetext	E
Subtype	Highlight	Annotation	highlight	E
Subtype	Ink	Annotation	ink	E
Subtype	Line	Annotation	line	E
Subtype	Link	Annotation	link	E
Subtype	Polygon	Annotation	polygon	E
Subtype	Polyline	Annotation	polyline	E
Subtype	Popup	Annotation	popup	E
Subtype	Projection	Annotation	projection	E
Subtype	Redact	Annotation	redact	E
Subtype	Sound	Annotation	sound	E
Subtype	Square	Annotation	square	E
Subtype	Squiggly	Annotation	squiggly	E
Subtype	Stamp	Annotation	stamp	E
Subtype	StrikeOut	Annotation	strikeout	E
Subtype	Text	Annotation	text	E
Subtype	Underline	Annotation	underline	E
Subtype		Embedded file stream	mimetype	A
Subtype		Mac OS file information	subtype	A
Sy		Caret annotation	symbol	A
T		Annotation	title	A
T		FDF field	name	E
top		Destination syntax	Top	A
URI		URI action	Name	A
URI		Action type	URI	E
V		FDF field	value	E
Vertices		Polygon or polyline annotation	vertices	E
W		Border style	width	A
XYZ		Destination syntax	XYZ	E
zoom		Destination syntax	Zoom	A

XFDF to PDF

This table shows the mapping between XFDF element or attribute and PDF key. The E/A column indicates whether the XFDF name corresponds to an element or attribute.

XFDF	E/A	PDF key	Key value	Dictionary
Action	E	A		Link annotation
annots	E	Annots		FDF
appearance	E	AP		Annotation
bits	A	B		Sound object
BorderStyleAlt	E	Border		Link annotation
Bottom	A	bottom		Destination syntax annotation
caption	A	Cap		Line annotation
caption-offset-h	A	CO		Line annotation
caption-offset-v	A	CO		Line annotation
caption-style	A	CP		Line annotation
caret	E	Subtype	Caret	Annotation
channels	A	C		Sound object
checksum	A	CheckSum		Embedded file parameter
circle	E	Subtype	Circle	Annotation
color	A	C		Annotation
contents	E	Contents		Annotation
contents-richtext	E	RC		Markup annotation
coords	A	QuadPoints		Text markup annotation
coords	A	QuadPoints		Link annotation
coords	A	QuadPoints		Redaction annotation
creation	A	CreationDate		Embedded file parameter
creationdate	A	CreationDate		Markup annotation
creator	A	Creator		Mac OS file information
dashes	A	D		Border style
DashPattern	A	Border		Annotation
data	E			
date	A	M		Annotation
defaultappearance	E	DA		Free text annotation
defaultappearance	E	DA		Redaction annotation
defaultstyle	E	DS		Free text annotation
Dest	E	Dest		Link annotation

encoding	A			
encoding	A	E		Sound object
end	A	L		Line annotation
f	E	F		FDF
field	E	Fields		FDF
fields	E	Fields		FDF
file	A	F		File specification
File	E	F		Remote go-to or launch annotation
fileattachment	E	Subtype	FileAttachment	Annotation
filter	A	Filter		Stream
Fit	E	Fit		Destination syntax
FitB	E	FitB		Destination syntax
FitBH	E	FitBH		Destination syntax
FitBV	E	FitBV		Destination syntax
FitH	E	FitH		Destination syntax
FitR	E	FitR		Destination syntax
FitV	E	FitV		Destination syntax
flags	A	F		Annotation
freetext	E	Subtype	FreeText	Annotation
fringe	A	RD		Caret, square or circle annotation
gesture	E	InkList		Ink annotation
GoTo	E	GoTo		Action type
GoToR	E	GoToR		Action type
HCornerRadius	A	Border		Annotation
head	A	LE		Line annotation
Highlight	A	H		Link annotation
highlight	E	Subtype	Highlight	Annotation
href	A	F		FDF
icon	A	Name		File attachment annotation
icon	A	Name		Rubber stamp annotation
icon	A	Name		Sound annotation
icon	A	Name		Text annotation
ids	E	ID		FDF
ink	E	Subtype	Ink	Annotation

inklist	E	InkList		Ink annotation
inreplyto	A	IRT		Markup annotation
intensity	A	I		Border effect
intent	A	IT		Freetext, line, or polyline annotation
interior-color	A	IC		Square or circle annotation
interior-color	A	IC		Redaction annotation
IsMap	A	IsMap		Action dictionary annotation
justification	A	Q		Free text annotation
justification	A	Q		Redaction annotation
Launch	E	Launch		Action type
leader-offset	A	LLO		Line annotation
leaderExtend	A	LLE		Line annotation
leaderLength	A	LL		Line annotation
Left	A	left		Destination syntax annotation
length	A	Length		Stream
line	E	Subtype	Line	Annotation
link	E	Subtype	Link	Annotation
mimetype	A	Subtype		Embedded file stream
mode	A			
modification	A	ModDate		Embedded file parameter
modified	A	ID		FDF
Name	A			Destination syntax
Name	A	N		Named action
Name	A	URI		URI action
name	A	NM		Annotation
name	E	T		FDF field
Named	E	Named		Action type
Named	E			Destination syntax
NewWindow	A	NewWindow		Remote go-to action annotation
NewWindow	A	NewWindow		Launch parameter annotation
OnActivation	E	A		Annotation
opacity	A	CA		Markup annotation
open	A	Open		Pop-up annotation
original	A	ID		FDF

OriginalName	A	F		File specification annotation
overlayappearance	E	RO		Redaction annotation
overlay-text	A	OverlayText		Redaction annotation
overlay-text-repeat	A	Repeat		Redaction annotation
Page	A	page		Destination syntax
page	A	Page		FDF file
polygon	E	Subtype	Polygon	Annotation
polyline	E	Subtype	Polyline	Annotation
popup	E	Subtype	Popup	Annotation
projection	E	Subtype	Projection	Annotation
rate	A	R		Sound object
rect	A	Rect		Annotation
redact	E	Subtype	Redact	Annotation
replyType	A	RT		Markup annotation
resource	E	ResFork		Mac OS file information
Right	A	right		Destination syntax
rotation	A	Rotate		Freetext and stamp annotations
size	A	Size		Embedded file parameter
sound	E	Subtype	Sound	Annotation
square	E	Subtype	Square	Annotation
squiggly	E	Subtype	Squiggly	Annotation
stamp	E	Subtype	Stamp	Annotation
start	A	L		Line annotation
state	A	State		Text annotation
statemodel	A	StateModel		Text annotation
strikeout	E	Subtype	StrikeOut	Annotation
style	A	S		Border style
subject	A	Subj		Markup annotation
subtype	A	Subtype		Mac OS file information
symbol	A	Sy		Caret annotation
tail	A	LE		Line annotation
text	E	Subtype	Text	Annotation
title	A	T		Annotation
Top	A	top		Destination syntax
underline	E	Subtype	Underline	Annotation

URI	E	URI		Action type
value	E	V		FDF field
value-richtext	E	RV		Fields containing variable text
VCornerRadius	A	Border		Annotation
vertices	E	Vertices		Polygon or polyline annotation
Width	A	Border		Annotation
width	A	W		Border style
xfdf	E	Root		FDF
XYZ	E	XYZ		Destination syntax
Zoom	A	zoom		Destination syntax

List of References

PDF Reference:

http://www.adobe.com/go/acrobatsdk_pdf_reference

Changes From Earlier Versions

Version 3.0:

The projection annotation element was added.

A Appendix: XFDF Definitions for Comments on 3D or Rich Media Annotations

Introduction

In Acrobat, comments can also be placed on 3D or rich media annotations. A major difference between 3D or rich media annotations and other page annotations is in their ability to define structured content. For example, a 3D annotation may contain several views of a 3D model or a rich media annotation may contain a video sequence consisting of many frames. When a comment is placed on a 3D or rich media annotation, the comment is specific to a 3D view for the 3D page annotation, or to a specific frame in a video for a rich media annotation.

Some additional context specific information needs to be kept with the comment in order to support the connection between a comment and the 3D or rich media annotation. When a comment is placed on 3D or rich media annotation, this extra information identifies both the 3D or rich media annotation and the view within that annotation that the comment is associated with. To support the ability to export and import these comments and to ensure that their association to a 3D or rich media annotation remains intact, this extra data content must be part of the XFDF specification.

Various Scenarios of Comments on a 3D Annotation

Let us consider three different situations where comments are placed on a 3D or rich media annotation. The first situation occurs when a 2D page comment is selected from the Comment & Markup menu and placed on top of a 3D view. In this case, the comment is expected to only appear when its associated 3D view is visible (or when the comment is selected from the comment or view pane). The second situation occurs when a 2D page comment is selected from the Comment & Markup menu and placed on top of a rich media view. In this case, the comment is expected to only appear when its associated video frame is visible. The third situation occurs when a 3D comment or measurement is placed on a 3D view. In this case, the comment's markup is a part of the 3D scene definition and a projection annotation is used as a surrogate within the commenting system to represent the 3D comment.

In all three cases, comment's extra data contains the information to identify comment's associated 3D or rich media annotation and the view within that annotation. One complication that arises is that when a comment is placed on a 3D or rich media annotation, it is possible that a new 3D view is also created. If the user creates a new view with a comment and exports that comment, it is expected that when the comment is imported into a document where the view does not exist yet, the view will be created in the importing document. To support this, the view definition contained within the extra data structure should contain enough information to accurately recreate the original view.

Example of a comment on a 3D Annotation

The following is an example XFDF file generated when a single 2D circle comment is placed on a 3D view and exported. The extra data-related tags are displayed in red.

```
<?xml version="1.0" encoding="UTF-8"?>
<xfdf xmlns="http://ns.adobe.com/xfdf/" xml:space="preserve">
 <annots>
 <circle color="#FF0000" creationdate="D:20081106121144-08'00'" flags="print"
       date="D:20081106121144-08'00'"
       name="bc74fb1b-5044-4691-8b5a-af9de0923719" page="0"
       fringe="0.5,0.5,0.5,0.5" rect="230.045395,452.626038,374.557587,
          548.992188"
          subject="Oval" title="jahall">
     <ex_data subtype="Markup3D">
      <exdata3d>
       <anno3dname>3D1</anno3dname>
       <view3d>
             View definition tags omitted
       </view3d>
      </exdata3d>
     </ex_data>
       <popup flags="print,nozoom,norotate" open="no" page="0"
          rect="612,428.492188,792,548.492188"/>
     </circle>
   </annots>
    <f href="vice_copy3.pdf"/><ids original="D0F211E86F278A45A4C62BE7B0DF7C9F"
       modified="E6832FBBAA8B5840B0E075B9D59E71D4"/>
  </xfdf>
```

The ex_data tag is a child of a specific type of annotation; in this case circle, and has an attribute that defines the type of extra data contained within. The type of data contained is a function of the context in which the comment appears. In this example, the context type is ‘Markup3D’ indicating that the content contains information identifying 3D annotation and defining the 3D view contents.

The ex_data Annotation Subelement

ex_data

The ex_data element is a child of an annotation element and serves as a container for extra context specific data elements. The ex_data element corresponds to the ExData key in the FDF dictionary. The ex_data annotation subelement is intended to be an open-ended hook for metadata inclusion that supports all present and future ExData subtypes. It is not specific to commenting on 3D annotations.

The ex_data subelement applies to all of the markup annotation types supported in XFDF. The various markup annotation types supported in XFDF are:

- Text
- FreeText
- Line
- Square
- Circle
- Polygon
- Polyline
- Highlight
- Underline
- Squiggly
- StrikeOut
- Stamp
- Caret

- Ink
- Popup
- FileAttachment
- Sound
- Projection

To ensure the security of imported XFDF data, no ex_data subelement nor any of its subelements, recursively, may refer to any external XML entity.

Content model

```
(exdata3d? | other_type_specific_content)
```

Attributes

`subtype` Required. The type of extra data contained corresponds to the PDF ExData dictionary's Subtype entry. Currently there are 4 subtypes supported:

- Markup3D - a 2D page comment placed on a 3D annotation.
- 3DM - a 3D comment or 3D measurement.
- RichMedia - a 2D page comment on a rich media annotation.
- GeoMarkup - a 2D measurement placed on 2D geospatial data. The GeoMarkup type contains no extra data but is used as a tag, such that the application knows that the comment is associated to 2D geospatial data.

The exdata3D Related Elements

exdata3d

The exdata3d element is a child of an ex_data element and serves as the top level container for data defining the connection between a comment and a 3D or rich media annotation. This element is valid for ex_data types: Markup3D, RichMedia, and 3DM. The measurementcomment comment tag is only valid in the context of the 3DM type.

Content model

```
((anno3dname | md5checksum ) & view3d & measurementcomment*)
```

Attributes

None.

anno3dname

The anno3dname element is a child of an exdata3d element and defines the name of the 3D or rich media annotation to which this comment is associated. The containing annotation element defines the page on which the comment is present; therefore annotation names only need to be unique within a single page. The anno3dname matches the NM key in the annotations dictionary.

Content model

Text String.

Attributes

None.

md5checksum

Note that all annotations are not named; the NM key is optional. So if the annotation is not named, the md5checksum tag is used to identify the appropriate 3D or rich media annotation. The md5checksum is the checksum of the data associated with the 3D or rich media annotation. The md5checksum element is a child of an exdata3d element.

Content model

Text String.

Attributes

None.

measurename

For 3D comments and measurements, the specific 3D markup within the view also needs to be identified. The view has a list of named measurements (or 3D comments); this tag will match one of the names on the view's list. This tag will only be present if the ex_data type is 3DM.

Content model

Text String.

Attributes

None.

The view3d Related Elements

For 3D annotations, a 3D view defines the set of parameters that will create a specific presentation of a collection of 3D scene geometry.

The following is an example of a 3D view definition.

```
<view3d>
 <externalname>SectionView5</externalname>
   <internalname>60abf029-e84c-4969-8359-18f1639c6077</internalname>
   <cameraxform>
    -0.493431 0.869785 0.000000 0.000000
    0.307474 0.174430 0.935433 0.000000
```

```
         0.813625 0.461571 -0.353505 0.000000
         -0.356615 -0.287046 0.181059 1.000000
        </cameraxform>
        <targetdistance>0.434344</targetdistance>
        <projection projtype="perspective">
         <fieldofview>30.000001</fieldofview>
         <scaletype>min</scaletype>
        </projection>
        <background entirebackground="true">0.937256 0.937256
        0.937256</background>
        <renderinginfo mode="solid"/>
        <lighting scheme="headlamp"/>
        <shouldresetnodes/>
    </view3d>
```

view3d

The view3d tag is the top container for the data required to define the 3D view that a comment is associated with.

Content model

```
(externalname & internalname & (cameraxform | u3dmatrixsource) &
targetdistance & projection & clip & background &
renderinginfo & lighting & crosssection? & nodeparameter* &
shouldresetnodes? & measure* & stateinfo* & snapshot?)
```

Attributes

None.

externalname

The externalname tag defines the user visible name for the view.

Content model

Text String.

Attributes

None.

Camera Related Elements

The camera transformation associated with a view can be specified by two methods. The first is by directly specifying the transformation matrix and the second is by passing the Id of a node in the U3D scene that contains a transformation matrix. For further information, please refer to the PDF specification’s 3D view definition.

cameraxform

The cameraxform tag defines a 12 element 3D transformation matrix that specifies the position and orientation of the camera in world coordinates.

Content model

A list of 12 doubles.

Attributes

None.

u3dmatrixsource

The u3dmatrixsource tag defines the name of a node in the U3D scene graph that contains a view transformation.

Content model

Text String.

Attributes

None.

targetdistance

The targetdistance tag defines a non-negative number indicating a distance in the camera coordinate system along the z-axis to the center of orbit for this view.

Content model

Double.

Attributes

None.

View Projection Related Elements

projection

The projection tag is a container whose entries define the mapping of 3D camera coordinates onto the target coordinate system of the annotation.

Content model

```
((fieldofview & (scalevalue | scaletype)? ) | (viewplanesize? & scaletype?))
```

Attributes

`projtype` Required. The type of view projection:

- parallel
- perspective

fieldofview

The fieldofview tag is valid only for perspective projections. It defines a number between 0 and 180, inclusive, specifying the field of view of the virtual camera, in degrees. It defines a cone in 3D space centered around the z-axis and a circle where the cone intersects the near clipping plane. The circle, along with the value formed from either the scalevalue or the scaletype, specify the scaling of the projected artwork when rendered in the 2D plane of the annotation.

Content model

Double.

Attributes

None.

viewplanesize

The viewplanesize tag is valid only for parallel projections. It defines a positive number that specifies the scale factor to be applied to both the *x* and *y* coordinates when projecting onto the annotation's target coordinate system.

Content model

Double.

Attributes

None.

scalevalue

The scalevalue tag is only valid for perspective projections. It defines the diameter of the circle formed by the intersection of the near plane and the cone specified by fieldofview. For parallel projections, it defines a positive number that specifies the scale factor to be applied to both the *x* and *y* coordinates when projecting onto the annotation's target coordinate system.

Content model

Double.

Attributes

None.

scaletype

The scaletype tag contains a name that specifies a strategy for binding (scaling to fit) the near plane's *x* and *y* coordinates onto the annotation's target coordinate system. For parallel projections, the scaling specified in this entry is applied in addition to the scaling factor specified by the viewplanesize entry.

Content model

Text string whose value may be one of the following:
W - Scale to fit the width of the annotation
H - Scale to fit the height of the annotation
Min - Scale to fit the lesser of width or height of the annotation
Max - Scale to fit the greater of width or height of the annotation
Absolute - No scaling should occur due to binding (parallel projections only)

Attributes

None.

clip

The clip tag is present when near and far clipping planes need to be applied to the view.

Content model

None.

Attributes

`Near`	Required for perspective projections; optional for parallel projections. The near clipping distance, expressed in the camera coordinate system. No parts of objects whose z-coordinates are less than the value of this entry are drawn. For perspective projections, the value must be positive; for parallel projections the value must be non-negative, and the default value is 0.
`Far`	Optional. The far clipping distance, expressed in the camera coordinate system. No parts of objects whose z-coordinates are greater than the value of this entry are drawn. If this entry is absent, no far clipping occurs.

View Background Related Elements

background

The background tag contains information about the how the view background should be rendered.

Content model

`(color)`

Attributes

`entirebackground`	If **true**, the background should apply to the entire annotation; if **false**, the background should apply only to the rectangle specified by the annotation’s 3D view box. The default value is **false**.

color

The color tag defines a 3 element list representing the RGB color for the background.

Content model

A list of 3 doubles.

Attributes

None.

Model Rendering Related Elements

renderinginfo

The renderinginfo tag contains information about how the view geometry should be rendered.

Content model

`(auxcolor? & facecolor? & opacity? & creasevalue?)`

Attributes

`mode`

- `solid` displays textured and lit geometric shapes.
- `solidwireframe` displays textured and lit geometric shapes (triangles) with single color edges on top of them. The color of these edges is determined by the **auxcolor** entry.
- `transparent` displays textured and lit geometric shapes (triangles) with an added level of transparency.
- `transparentwireframe` displays textured and lit geometric shapes (triangles) with an added level of transparency, with single color opaque edges on top of it. The color of these edges is determined by the **auxcolor** entry.
- `boundingbox` displays the bounding box edges of each node, aligned with the axes of the local coordinate space for that node. The color of the bounding box edges is determined by the **auxcolor** entry.
- `transparentboundingbox` displays bounding boxes faces of each node, aligned with the axes of the local coordinate space for that node, with an added level of transparency. The color of the bounding box faces is determined by the **facecolor** entry.
- `transparentboundingboxoutline` displays bounding boxes edges and faces of each node, aligned with the axes of the local coordinate space for that node, with an added level of transparency. The color of the bounding box edges is determined by the **auxcolor** entry. The color of the bounding boxes faces is determined by the **facecolor** entry.
- `wireframe` displays only edges in a single color. The color of these edges is determined by the **auxcolor** entry.
- `shadedwireframe` displays only edges; although interpolates their color between their two vertices and applies lighting.
- `hiddenwireframe` displays edges in a single color; although removes back-facing and obscured edges. The color of these edges is determined by the **auxcolor** entry.
- `vertices` displays only vertices in a single color. The color of these points is determined by the **auxcolor** entry.
- `shadedvertices` displays only vertices; although uses their vertex color and applies lighting.
- `illustation` displays silhouette edges with surfaces and removes obscured lines. The color of these edges is determined by the **auxcolor** entry, and the color of the surfaces is determined by the **facecolor** entry.
- `solidoutline` displays silhouette edges with lit and textured surfaces and removes obscured lines. The color of these edges is determined by the **auxcolor** entry.
- `shadedillustation` displays silhouette edges with lit and textured surfaces and an additional emissive term to remove poorly lit areas of the artwork. The color of these edges is determined by the **auxcolor** entry.

auxcolor

The auxcolor tag defines a 3 element list representing the RGB used in many of the rendering modes.

Content model

A list of 3 doubles.

Attributes

None.

facecolor

The facecolor tag defines a 3 element list representing the RGB color used for illustration and transparent bounding box outline rendering modes.

Content model

A list of 3 doubles.

Attributes

None.

opacity

The opacity tag defines a number specifying the opacity applied by some render modes, using a standard additive blend. The default value is 0.5.

Content model

Double.

Attributes

None.

creasevalue

The creasevalue tag contains a number specifying the angle, in degrees, to be used as the crease value for determining silhouette edges. If two front-facing faces share an edge and the angle between the normals of those faces is greater than or equal to the crease value, the shared edge is considered to be a silhouette edge. The default value is 45.

Content model

Double.

Attributes

None.

Lighting Related Elements

lighting

The lighting tag specifies the lighting scheme to be used when rendering 3D artwork with this view.

Content model

None.

Attributes

`scheme`

- `artwork` - lights as specified in the 3D artwork
- `none` - no lights are used
- `white` - three blue-grey infinite lights, no ambient term

  Light 1Color: < 0.38, 0.38, 0.45 > Direction: < -2.0, -1.5, -0.5 >

  Light 2Color: < 0.6, 0.6, 0.67 > Direction: < 2.0, 1.1, -2.5 >

  Light 3Color: < 0.5, 0.5, 0.57 > Direction: < -0.5, 0.0, 2.0 >
- `day` - three light-grey infinite lights, no ambient term

  Light 1Color: < 0.5, 0.5, 0.5 > Direction: < -2.0, -1.5, -.5 >

  Light 2Color: < 0.8, 0.8, 0.9 > Direction: < 2.0, 1.1, -2.5 >

  Light 3Color: < 0.9, 0.9, 0.9 > Direction: < 0.02, 0.01, 2.0 >
- `night` - one yellow, one aqua, and one blue infinite light, no ambient term

  Light 1Color: < 1, .75, .39 > Direction: < -2.0, -1.5, -0.5 >

  Light 2Color: < 0.31, 0.47, 0.55 > Direction: < 2.0, 1.1, -2.5 >

  Light 3Color: < .5, .5, 1.0 > Direction: < 0.0, 0.0, 2.0 >
- `hard` - three grey infinite lights, moderate ambient term

  Light 1Color: < 0.5, 0.5, 0.5 > Direction: < -1.5, -1.5, -1.5 >

  Light 2Color: < 0.8, 0.8, 0.9 > Direction: < 1.5, 1.5, -1.5 >

  Light 3Color: < 0.9, 0.9, 0.9 > Direction: < -0.5, 0, 2.0 >

  AmbientColor: < 0.5, 0.5, 0.5 >
- `primary` - one red, one green, and one blue infinite light, no ambient term

  Light 1Color: < 1, 0.2, 0.5 > Direction: < -2, -1.5, -0.5 >

  Light 2Color: < 0.2, 1.0, 0.5 > Direction: < 2.0, 1.1, -2.5 >

  Light 3Color: < 0, 0, 1 > Direction: < 0.0, 0.0, 2.0 >

`scheme`	● `blue` - three blue infinite lights, no ambient term Light 1Color: < 0.4, 0.4, 0.7 > Direction: < -2.0, -1.5, -0.5 > Light 2Color: < 0.75, 0.75, 0.95 > Direction: < 2.0, 1.1, -2.5 > Light 3Color: < 0.7, 0.7, 0.95 > Direction: < 0.0, 0.0, 2.0 > ● `red` - three red infinite lights, no ambient term Light 1Color: < 0.8, 0.3, 0.4 > Direction: < -2.0, -1.5, -0.5 > Light 2Color: < 0.95, 0.5, 0.7 > Direction: < 2.0, 1.1, -2.5 > Light 3Color: < 0.95, 0.4, 0.5 > Direction: < 0.0, 0.0, 2.0 > ● `cube` - six grey infinite lights aligned with the major axes, no ambient term Light 1Color: < .4, .4, .4 > Direction: < 1.0, 0.01, 0.01 > Light 2Color: < .4, .4, .4 > Direction: < 0.01, 1.0, 0.01 > Light 3Color: < .4, .4, .4 > Direction: < 0.01, 0.01, 1.0 > Light 4Color: < .4, .4, .4 > Direction: < -1.0, 0.01, 0.01 > Light 5Color: < .4, .4, .4 > Direction: < 0.01, -1.0, 0.01 > Light 6Color: < .4, .4, .4 > Direction: < 0.01, 0.01, -1.0 > ● `cad` - three grey infinite lights and one light attached to the camera, no ambient term Light 1Color: < 0.72, 0.72, 0.81 > Direction: < 0.0, 0.0, 0.0 > Light 2Color: < 0.2, 0.2, 0.2 > Direction: < -2.0, -1.5, -0.5 > Light 3Color: < 0.32, 0.32, 0.32 > Direction: < 2.0, 1.1, -2.5 > Light 4Color: < 0.36, 0.36, 0.36 > Direction: < 0.04, 0.01, 2.0 > ● `headlamp` - single infinite light attached to the camera, low ambient term Light 1Color: < 0.8, 0.8, 0.9 > Direction: < 0.0, 0.0, 0.0 > AmbientColor: < 0.1, 0.1, 0.1 >

Cross Section Related Elements

The following is an example of a cross section definition.

```
<crosssection>
  <centerofrotation>0.000000 -0.093300 0.026100</centerofrotation>
  <planetilt1>0.000000</planetilt1>
  <planetilt2>0.000000</planetilt2>
  <alignment axis="z"/>
  <intersectionsvisible>
    <planecolor>0.999985 0.000000 0.000000</planecolor>
  </intersectionsvisible>
</crosssection>
```

crosssection

The crosssection tag is a container whose entries define the parameters for a section plane operation to be applied to the view.

Content model

```
(centerofrotation? & planetilt1, planetilt2 & alignment & planevisible? &
intersectionsvisible?)
```

Attributes

`planeflip` If **true**, the geometry on the negative side of the section plane (w.r.t. the plane normal) will be clipped, otherwise geometry on the positive side of the section plane will be clipped.

`showtransparent` If **true**, the clipped geometry will be rendered in a transparent rendering mode.

centerofrotation

The centerofrotation tag contains three doubles specifying the (x, y, z) position of the center of rotation on the cutting plane in world space coordinates.

Content model

A list of three doubles.

Attributes

None.

planetilt1, planetilt2

The section plane orientation is defined by specifying two angles (planetilt1, planetilt2) in degrees relative to either the x, y, or z axis.

Content model

Double.

Attributes

None.

alignment

The axis to which the tilt angles are relative.

Content model

Text string; valid values are *x*, *y*, or *z*.

Attributes

None.

planevisible

The planevisible tag, when present, indicates that the section plane should be visible in the 3D scene. The contents of this tag control the planes appearance.

Content model

```
(planecolor? & planeopacity?)
```

Attributes

None.

planecolor

The planecolor tag defines a 3 element list representing the RGB color used in rendering the section plane.

Content model

A list of three doubles.

Attributes

None.

planeopacity

The planeopacity tag defines a number specifying the opacity of the section plane. The default value is 0.5.

Content model

Double.

Attributes

None.

intersectionsvisible

The intersectionsvisible tag, when present, indicates that the lines of intersection between the scene geometry and the section plane should be visible in the 3D scene. The contents of this tag control the planes appearance.

Content model

```
(intersectioncolor?)
```

Attributes

None.

intersectioncolor

The intersectioncolor tag defines a 3 element list representing the RGB color used in rendering the lines of intersection between the scene geometry and the section plane.

Content model

A list of 3 doubles.

Attributes

None.

View Specific Node Control Related Elements

nodeparameter

The nodeparameter tag is a container for view specific attributes to be applied to specific nodes in a 3D scene. Note that parent view3d tag can contain many nodeparameter tags, defining parameters for different nodes in the scene.

Content model

```
(nodeid & nodexform? & opacity? & noderendermode?)
```

Attributes

`visible`	Boolean value defining the nodes visibility.

nodeid

The nodeid tag defines the name of a node in the scene graph that the node parameters are to be applied to.

Content model

Text String.

Attributes

None.

nodexform

The nodexform tag defines a transformation matrix that is to be applied to the named node in the scene graph.

Content model

A list of 12 doubles.

Attributes

None.

opacity

The opacity tag defines an opacity value that is to be applied to the named node in the scene graph.

Content model

Double.

Attributes

None.

noderendermode

The noderendermode tag defines a render mode value that is to be applied to the named node in the scene graph.

Content model

Text String is constrained to the values listed in the renderinginfo tag.

```
“solid”, “solidwireframe”, “transparent”, “transparentwireframe”,
“boundingbox”, “transparentboundingbox”, “transparentboundingboxoutline”,
“wireframe”, “shadedwireframe”, “hiddenwireframe”, “vertices”,
“shadedvertices”, “illustration”, “solidoutline”, “shadedillustration”
```

Attributes

None.

Rich Media Related Elements

stateinfo

The stateinfo tag contains a text string containing state data to be passed to a rich media instance (usually a player) when the view is triggered. For example, an instance might contain a string that when sent to the player will move to a specific time with a video.

Content model

Text String.

Attributes

`instance`	Required Text String; the name of the rich media instance.

snapshot

The snapshot tag contains an image that is displayed when the view is activated. If this tag is present, the image it describes is displayed in place of the current artwork.

Content model

Text String; the image is defined as base 64 encoded string.

Attributes

None.

Measurement Related Elements

measure

The measure tag is the top level container for all 3D measurement and comment instances. A 3D view may contain a list of one or more 3D measurements or 3D comments that are associated with the view content, so there may be many instances of the measure tag within a parent view3d tag.

Content model

```
(measurename & (linearmarkup | perpendicularmarkup |angularmarkup |
radialmarkup | comment3dmarkup))
```

Attributes

None.

measurename

The measurename tag defines the user visible name for the 3D measurement or comment.

Content model

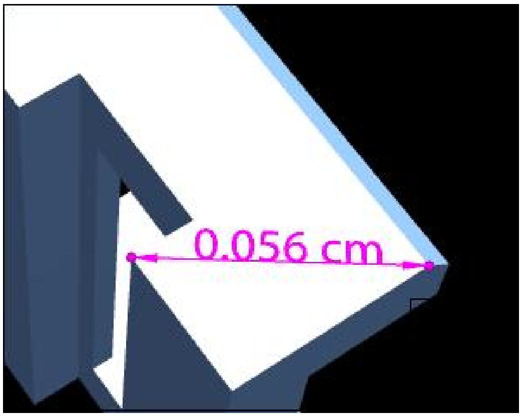
Text String.

Attributes

None.

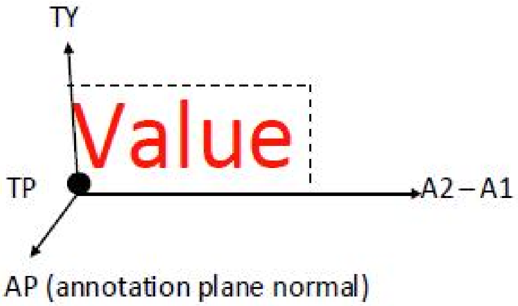
Linear Dimension Related Elements

A linear measurement is a markup showing the distance between two arbitrary points on a 3D model. An example is displayed in the following figure.



As displayed in the figure above, a 3D linear measurement consists of two filled circles (called anchor points), one at each of the two positions being measured, and a line with an arrowhead on each end connecting the two anchor points (referred to as the measure line). The measure line is labeled with a value representing the distance between the two anchor points.

As displayed in the figure below, the value text is drawn on the Annotation Plane (AP) where the horizontal text path is defined by the vector from Anchor Point 1 (A1) to Anchor Point 2 (A2) with the text up direction defined by the vector TY. The lower left corner of the text box is positioned at the text anchor point (TP).



The actual Y axis is formed by taking the cross product of AP and (A2 – A1); the vector TY is only used to determine the orientation of this Y axis.

If the text position TP is outside the area between A1 and A2, an extension line collinear to the measure line connecting TP to the closest anchor point is generated.

There are three parts to the text string displayed with the measurement; a numeric value, a units string, and an optional user string. The display of the numeric value field number is also controlled by the precision value, which indicates how many digits to the right of the decimal point should be displayed. The viewer can convert the numeric value to a string and combine it with the units string and user text as appropriate.

The following is an example of a linear dimension.

```
<measure>
 <measurename>Measurement1</measurename>
 <linearmarkup>
      <annoplane> -0.981099 0.122982 0.149397</annoplane>
      <anchor1>-0.051400 -0.019600 0.052200</anchor1>
      <anchor1partname>Base</anchor1partname>
      <anchor2>-0.050300 -0.065700 0.052100</anchor2>
      <anchor2partname>Base</anchor2partname>
      <textposition>-0.050994 -0.036613 0.052163</textposition>
      <textydirection>0.150717 0.001452 0.988576</textydirection>
      <textsize>20.000000</textsize>
      <markupcolor>0.000000 1.000000 0.000000</markupcolor>
      <value>0.056113</value>
      <units>cm</units>
      <precision>3</precision>
 </linearmarkup>
</measure>
```

linearmarkup

The linearmarkup tag is a container that holds the data defining an instance of a point to point measurement.

Content model

```
(annoplane & anchor1 & anchor1partname? & anchor2 & anchor2partname? &
textposition & textydirection & textsize? & markupcolor? & value & units &
precision? & usertext?)
```

Attributes

None.

annoplane

The annoplane tag contains a list of three doubles defining the normal for the 3D annotation plane on which the measurement markup will lie.

Content model

A list of three doubles.

Attributes

None.

anchor1

The anchor1 tag contains a list of three doubles defining the model space position of the first anchor point in world space. It is assumed that this is a position on the 3D model associated with this view.

Content model

A list of three doubles.

Attributes

None.

anchor1partname

The anchor1partname tag defines the node name for the part that the point anchor1 lies on.

Content model

Text String.

Attributes

None.

anchor2

The anchor2 tag contains a list of three doubles defining the model space position of the second anchor point in world space. It is assumed that this is a position on the 3D model associated with this view.

Content model

A list of three doubles.

Attributes

None.

anchor2partname

The anchor2partname tag defines the node name for the part that the point anchor2 lies on.

Content model

Text String.

Attributes

None.

textposition

The textposition tag contains a list of three doubles defining the model space position of the text anchor (the start of the text string) associated with the measurement.

Content model

A list of three doubles.

Attributes

None.

textydirection

The textydirection tag contains a list of three doubles defining the up direction vector for the text string presenting the measurement value.

Content model

A list of three doubles.

Attributes

None.

textsize

The textsize tag contains the size for the text string presenting the measurement value. Note that the text associated with 3D measurements and comments scales with the view; so the text size is defined as the size in inches when the view is first activated.

Content model

Double.

Attributes

None.

markupcolor

The markupcolor tag contains a list of three doubles defining the RGB color used in presenting the measurement markup.

Content model

A list of three doubles.

Attributes

None.

value

The value tag defines the numerical value representing a measurement value. This value is converted to a text string and displayed as part of the measurement text string.

Content model

Double.

Attributes

None.

units

The units tag contains a text string indicating the units of the associated measure value. This is appended to the value string when presenting the measurement.

Content model

Text String.

Attributes

None.

precision

The precision tag defines the number of decimal digits shown for the measurement value. If not specified, the default is three decimal places.

Content model

Integer.

Attributes

None.

usertext

The usertext tag contains an optional text string that is appended to the end of the measurement.

Content model

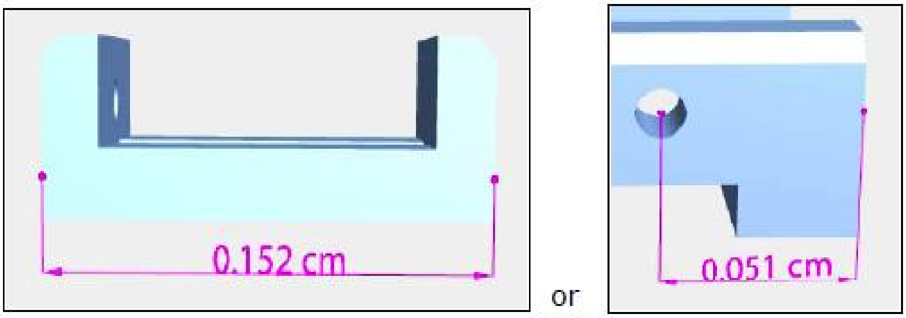
Text String.

Attributes

None.

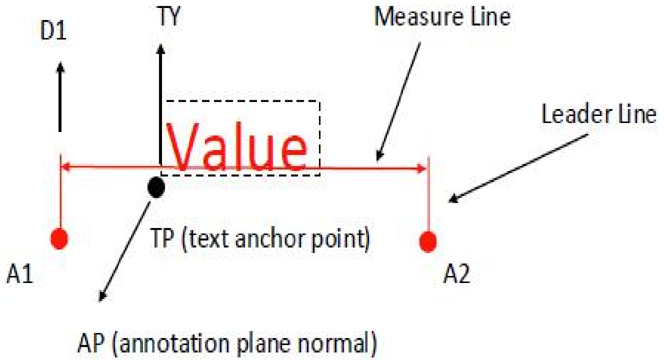
Perpendicular Dimension Related Elements

A perpendicular measurement is used to denote the perpendicular distance between two geometric entities (normally two lines or a point and a line), as displayed in the following figure.



As illustrated in the figures above, a perpendicular measurement markup consists of two filled circles at the anchor points, and two parallel extension lines (referred to as leader lines) that start at the anchor points and extend away from the anchor points. There is also a labeled line with arrowheads on both sides (referred to as the measure line), which indicates that the distance shown is the perpendicular distance between the two parallel lines.

The following figure displays the parameters associated with the perpendicular dimension.



In the above figure, red items are the measure markup and black items are parameters. All markup items are drawn on the Annotation Plane (AP). The text layout is defined in the similar manner as for linear dimensions. The lower left corner of the text box is positioned at the Text Anchor Point (TP) and the text's X axis is aligned with the measure line and the text flows in the direction defined by a vector from Anchor Point 1 (A1) to Anchor Point 2 (A2). The text's up direction is defined as the cross product of the annotation plane normal and the text X axis, in the direction defined by the TY parameter.

In addition to controlling text position, the Text Anchor Point (TP) also controls the lengths of leader lines and the placement of the measure line. Since leader lines are parallel and the measure line must be perpendicular to both leader lines, the intersection of leader lines and the measure line is easily computed.

If the text position TP is outside the area between A1 and A2, an extension line collinear to the measure line connecting TP to the closest anchor point is generated.

There are three parts to the text string displayed with the measurement; a numeric value, a units string, and an optional user string. The display of the numeric value field number is also controlled by the precision value, which indicates how many digits to the right of the decimal point should be displayed. The viewer can convert the numeric value to a string and combine it with the units string and user text as appropriate.

The following is an example of a perpendicular measurement.

```
<measure>
 <measurename>Measurement1</measurename>
 <perpendicularmarkup>
      <annoplane> -1.000000 -0.000000 -0.000000</annoplane>
      <anchor1>-0.051400 0.000000 0.000000</anchor1>
      <anchor1partname>Base</anchor1partname>
      <anchor2>-0.051400 -0.152400 0.000000</anchor2>
      <leaderdirection>0.000000 -0.000000 -1.000000</leaderdirection>
      <textposition>-0.051400 -0.057428 -0.017686</textposition>
      <textydirection>-0.000000 -0.000000 1.000000</textydirection>
      <textsize>20.000000</textsize>
      <markupcolor>0.000000 1.000000 0.000000</markupcolor>
      <value>0.152400</value>
      <units>cm</units>
      <precision>3</precision>
 </perpendicularmarkup>
</measure>
```

perpendicularmarkup

The perpendicularmarkup tag is a container that holds the data defining an instance of a perpendicular measurement between two geometric items.

Content model

```
(annoplane & anchor1 & anchor1partname? & anchor2 & anchor2partname? &
leaderdirection & textposition & textydirection & textsize? & markupcolor? &
value & units & precision? & usertext?)
```

Attributes

None.

annoplane

The annoplane tag contains a list of three doubles defining the normal for the 3D annotation plane on which the measurement markup will lie.

Content model

A list of three doubles.

Attributes

None.

anchor1

The anchor1 tag contains a list of three doubles defining the model space position of the first anchor point in world space. It is assumed that this is a position on the 3D model associated with this view.

Content model

A list of three doubles.

Attributes

None.

anchor1partname

The anchor1partname tag defines the node name for the part that the point anchor1 lies on.

Content model

Text String.

Attributes

None.

anchor2

The anchor2 tag contains a list of three doubles defining the model space position of the second anchor point in world space. It is assumed that this is a position on the 3D model associated with this view.

Content model

A list of three doubles.

Attributes

None.

anchor2partname

The anchor2partname tag defines the node name for the part that the point anchor2 lies on.

Content model

Text String.

Attributes

None.

leaderdirection

The leaderdirection tag contains a list of three doubles defining the model space direction of the leader lines associated with this perpendicular dimension.

Content model

A list of three doubles.

Attributes

None.

textposition

The textposition tag contains a list of three doubles defining the model space position of the text anchor (the start of the text string) associated with the measurement.

Content model

A list of three doubles.

Attributes

None.

textydirection

The textydirection tag contains a list of three doubles defining the up direction vector for the text string presenting the measurement value.

Content model

A list of three doubles.

Attributes

None.

textsize

The textsize tag contains the size for the text string presenting the measurement value. Note that the text associated with 3D measurements and comments scales with the view; so the text size is defined as the size in inches when the view is first activated.

Content model

Double.

Attributes

None.

markupcolor

The markupcolor tag contains a list of three doubles defining the RGB color used in presenting the measurement markup.

Content model

A list of three doubles.

Attributes

None.

value

The value tag defines the numerical value representing a measurement value. This value is converted to a text string and displayed as part of the measurement text string.

Content model

Double.

Attributes

None.

units

The units tag contains a text string indicating the units of the associated measure value. This is appended to the value string when presenting the measurement.

Content model

Text String.

Attributes

None.

precision

The precision tag defines the number of decimal digits shown for the measurement value. If not specified, the default is three decimal places.

Content model

Integer.

Attributes

None.

usertext

The usertext tag contains an optional text string that is appended to the end of the measurement.

Content model

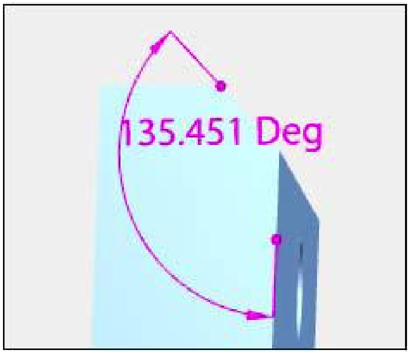
Text String.

Attributes

None.

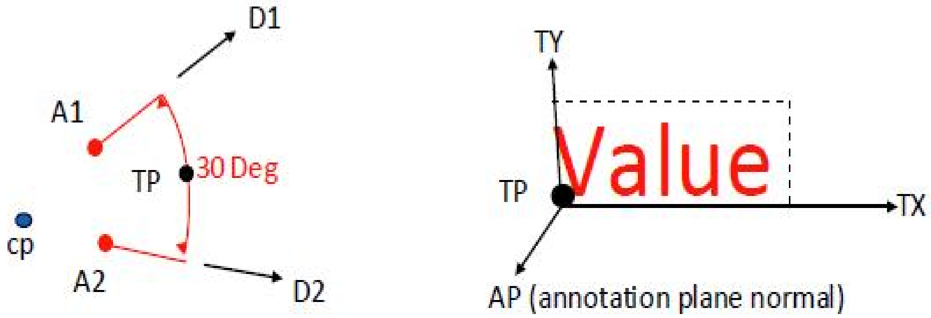
Angular Dimension Related Elements

An angular measurement is used to denote the angle between two linear entities, as displayed in the following figure.



As illustrated in the figure above, angular measurement markup consists of two anchor points, one located on each of the two linear entities whose angle is being measured. An extension line is connected to each anchor point, which is collinear with the edge it measures. A labeled arc, with an arrowhead at each end, connects the two extension lines to specify the angle being measured.

The following figure displays the key geometric parameters for an angular dimension.



The angle is defined by the measurement value (30 degree in the figure above) and is the angle between the leader direction vectors (D1 and D2). The angular measurement markup is generated by first computing the center point of the angle, cp in the figure above. Since the anchor points (A1 and A2) are on the annotation plane finding the intersection of the direction vectors (D1 and D2) passing through anchor points (A1 and A2) is a straightforward process. The text position (TP) controls the position of the measurement text, the placement of the angle arc, and the length and direction of the extension lines. The extensions lines are drawn from the anchor point to a point at a distance ||TP-cp|| from the center point cp along the associated direction vector (which is the intersection of the angle arc and the extension line). The angle arc center is at the center point cp (and its radius is ||TP – cp||) and is drawn between the two extensions lines. The markup text is displayed (based on the text orientation parameters) with the lower left corner of the text string starting at the text position (TP).

The text layout is defined in the similar manner as for other dimensions. The lower left corner of the text box is positioned at the text anchor point (TP) and the text's X axis is defined by the vector TX. The vector TX is expected to be orthogonal with the annotation plane normal. The text's up direction is defined as the cross product of the annotation plane normal and the text X axis, in the direction defined by the TY parameter.

The measurement value is interpreted as either being in degrees or radians as defined by the (DR) value and the appropriate label string is created.

There are three parts to the text string displayed with the measuremen;, a numeric value, a degree, or radians string and an optional user string. The display of the numeric value field number is also controlled by the precision value, which indicates how many digits to the right of the decimal point should be displayed. The viewer can convert the numeric value to a string and combine it with the degrees or radians string and user text as appropriate; this process is viewer dependent.

Special cases:

- Parallel direction vectors are invalid and no markup is generated.
- If the text position TP is outside the cone of the angle, an extension line is added to connect the text with the angle arc.

The following is an example of an angular measurement.

```
<measure>
 <measurename>Measurement1</measurename>
 <angularmarkup>
      <annoplane>-1.000000 0.000000 0.000000</annoplane>
      <anchor1>-0.051400 -0.146100 0.052200</anchor1>
      <leaderdirection1>0.000000 0.701517 0.712652</leaderdirection1>
      <anchor1partname>Base</anchor1partname>
      <anchor2>-0.051400 -0.152400 0.036819</anchor2>
      <leaderdirection2>0.000000 0.000000 -1.000000</leaderdirection2>
      <anchor2partname>Base</anchor2partname>
      <textposition>-0.051400 -0.136501 0.047616</textposition>
      <textxdirection>0.000000 -0.707953 -0.706260</textxdirection>
      <textydirection>-0.000000 -0.706260 0.707953</textydirection>
      <textsize>20.000000</textsize>
      <markupcolor>0.000000 1.000000 0.000000</markupcolor>
      <value>135.451096</value>
      <precision>3</precision>
 </angularmarkup>
</measure>
```

angularmarkup

The angularmarkup tag is a container that holds the data defining an instance of a angular measurement between two linear geometric items.

Content model

```
(annoplane & anchor1 & anchor1partname? & leaderdirection1 & anchor2 &
anchor2partname? & leaderdirection2 & textposition & textydirection &
textsize? & markupcolor? & value & units & precision? & usertext?)
```

Attributes

`radians`	Optional Boolean. If **true**, the angle measure is defined in radians. The default is **false** (angle in degrees).

annoplane

The annoplane tag contains a list of three doubles defining the normal for the 3D annotation plane on which the measurement markup will lie.

Content model

A list of three doubles.

Attributes

None.

anchor1

The anchor1 tag contains a list of three doubles defining the model space position of the first anchor point in world space. It is assumed that this is a position on the 3D model associated with this view.

Content model

A list of three doubles.

Attributes

None.

anchor1partname

The anchor1partname tag defines the node name for the part that the point anchor1 lies on.

Content model

Text String.

Attributes

None.

leaderdirection1

The leaderdirection1 tag contains a list of three doubles defining the model space direction of the leader line associated with anchor point 1.

Content model

A list of three doubles.

Attributes

None.

anchor2

The anchor2 tag contains a list of three doubles defining the model space position of the second anchor point in world space. It is assumed that this is a position on the 3D model associated with this view.

Content model

A list of three doubles.

Attributes

None.

anchor2partname

The anchor2partname tag defines the node name for the part that the point anchor2 lies on.

Content model

Text String.

Attributes

None.

leaderdirection2

The leaderdirection2 tag contains a list of three doubles defining the model space direction of the leader line associated with anchor point 2.

Content model

A list of three doubles.

Attributes

None.

textposition

The textposition tag contains a list of three doubles defining the model space position of the text anchor (the start of the text string) associated with the measurement.

Content model

A list of three doubles.

Attributes

None.

textxdirection

The textxdirection tag contains a list of three doubles defining the baseline direction vector for the text string presenting the measurement value.

Content model

A list of three doubles.

Attributes

None.

textydirection

The textydirection tag contains a list of three doubles defining the up direction vector for the text string presenting the measurement value.

Content model

A list of three doubles.

Attributes

None.

textsize

The textsize tag contains the size for the text string presenting the measurement value. Note that the text associated with 3D measurements and comments scales with the view; so the text size is defined as the size in inches when the view is first activated.

Content model

Double.

Attributes

None.

markupcolor

The markupcolor tag contains a list of three doubles defining the RGB color used in presenting the measurement markup.

Content model

A list of three doubles.

Attributes

None.

value

The value tag defines the numerical value representing a measurement value. This value is converted to a text string and displayed as part of the measurement text string.

Content model

Double.

Attributes

None.

units

The units tag contains a text string indicating the units of the associated measure value. This is appended to the value string when presenting the measurement.

Content model

Text String.

Attributes

None.

precision

The precision tag defines the number of decimal digits shown for the measurement value. If not specified, the default is three decimal places.

Content model

Integer.

Attributes

None.

usertext

The usertext tag contains an optional text string that is appended to the end of the measurement.

Content model

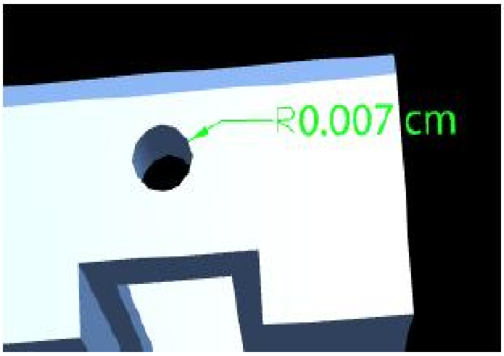
Text String.

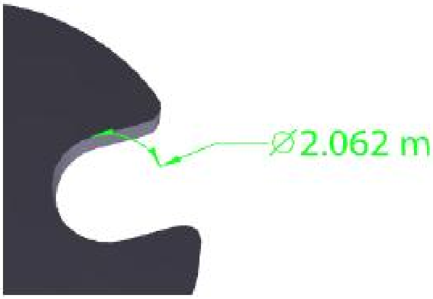
Attributes

None.

Radial Dimension Related Elements

The radial measurement is used to define the radius or diameter of a circular 3D entity. The following figures display two examples of a radial dimension.

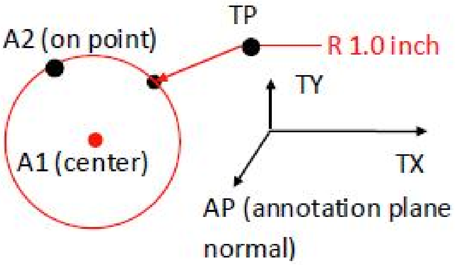




As illustrated in the left hand figure above, the basic markup for a radial dimension consists of an arrow pointing to a circle or arc connected to a leader line and text label defining the radius or diameter. If the arrow is positioned such that it is off the underlying arc, as displayed in the right hand figure, an extension arc is generated clarifying which arc is being measured.

For radius measurements, the measure value will be preceded by an ‘R’ in the measure string; and for diameter values, the measure value will be preceded by a Greek ‘phi’ symbol.

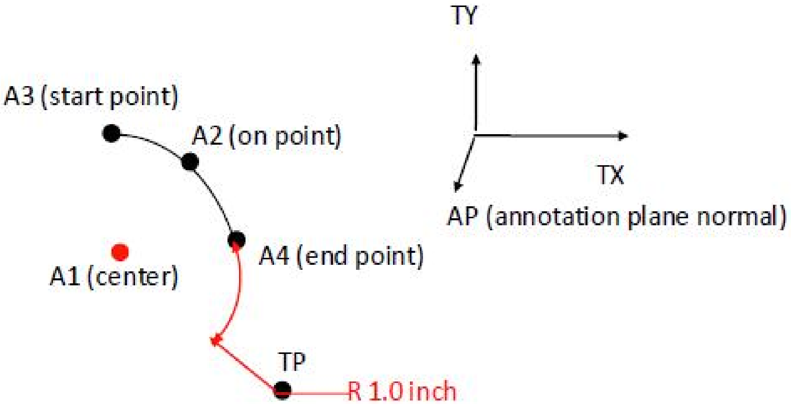
The parameters for defining the radial measurement for a circle are displayed in the figure below.



The circle being measured is defined by two points lying on the annotation plane, a circle center A1, and a point on the circle A2. For radial measurement, the text position (TP) controls the text string position, the arrow line orientation, and the extension line. The arrow line is drawn from the text position (TP) to the intersection point between the circle and a line from the text position to the circle center. The extension line is drawn from the text position (TP) in the direction of the text string's X axis (TX); and the length of the extension line is defined by the EL parameter. The left center of the measure value string begins at the end of the extension line. Vector TX is expected to be orthogonal with the annotation plane normal. The text's up direction is defined as the cross product of the annotation plane normal and the text's X axis, in the direction defined by the TY parameter.

There are three parts to the text string displayed with the measurement; a numeric value, a units string, and an optional user string. The display of the numeric value field number is also controlled by the precision value, which indicates how many digits to the right of the decimal point should be displayed. The viewer can convert the numeric value to a string and combine it with the units string and user text as appropriate; this process is viewer dependent.

The parameters for defining a radial dimension for an arc are very similar to that used for a circle, as displayed in the following figure.



The key difference for an arc is that there are two additional points defining the start and end position for the arc. In addition for the arc, it is possible for the text position to be defined, such that an extension arc having arrowhead on either end needs to be generated.

There is an optional parameter 'showcircle' that defines whether the underlying model circle or arc should be redrawn in the markup color to emphasize what is being measured.

The following is an example of an instance of a radial dimension.

```
<measure>
 <measurename>Measurement1</measurename>
 <radialmarkup>
      <annoplane>-0.006086 -0.999982 0.000000</annoplane>
      <circlecenter>-0.000191 -0.186570 0.033149</circlecenter>
      <pointoncircle>-0.008900 -0.186500 0.029300</pointoncircle>
      <arcstart>0.004600 -0.186600 0.041400</arcstart>
      <arcend>0.009100 -0.186600 0.031200</arcend>
      <anchorpartname>Shaft</anchorpartname>
      <textposition>0.000392 -0.186573 0.064191</textposition>
      <textxdirection>0.881165 -0.005363 -0.472779</textxdirection>
      <textydirection>0.472770 -0.002877 0.881181</textydirection>
      <textsize>20.000000</textsize>
      <markupcolor>0.000000 1.000000 0.000000</markupcolor>
      <value>0.009541</value>
      <units>cm</units>
      <precision>3</precision>
      <extensionlength>36.000000</extensionlength>
 </radialmarkup>
</measure>
```

radialmarkup

The radialmarkup tag is a container that holds the data defining an instance of a radial measurement of circular geometry.

Content model

```
(annoplane & circlecenter & pointoncircle & arcstart? & arcend? &
anchorpartname? & textposition & textydirection & textsize? & markupcolor? &
value & units & precision? & usertext? & extensionlength?)
```

Attributes

`showcircle` Optional Boolean. If **true**, will display the underlying circle associated with a radial dimension. The default is not to show the circle.

`diameter` Optional Radius/Diameter indicator. If **true**, the measurement value associated with a radial measurement represents a diameter as opposed to a radius value. The default is Radius.

annoplane

The annoplane tag contains a list of three doubles defining the normal for the 3D annotation plane on which the measurement markup will lie.

Content model

A list of three doubles.

Attributes

None.

circlecenter

The circlecenter tag contains a list of three doubles defining the model space position of the center of the circle or arc being measured.

Content model

A list of three doubles.

Attributes

None.

pointoncircle

The pointoncircle tag contains a list of three doubles defining the model space position of a point on the circle being measured.

Content model

A list of three doubles.

Attributes

None.

arcstart

If the geometry being measured is an arc, the arcstart tag contains a list of three doubles defining the model space position of the arc starting point in world space. It is assumed that this is a position on the 3D model associated with this view.

Content model

A list of three doubles.

Attributes

None.

arcend

If the geometry being measured is an arc, the arcend tag contains a list of three doubles defining the model space position of the arc ending point in world space. It is assumed that this is a position on the 3D model associated with this view.

Content model

A list of three doubles.

Attributes

None.

anchorpartname

The anchorpartname tag defines the node name for the part that the circle or arc lies on.

Content model

Text String.

Attributes

None.

textposition

The textposition tag contains a list of three doubles defining the model space position of the text anchor (the start of the text string) associated with the measurement.

Content model

A list of three doubles.

Attributes

None.

textxdirection

The textxdirection tag contains a list of three doubles defining the baseline direction vector for the text string presenting the measurement value.

Content model

A list of three doubles.

Attributes

None.

textydirection

The textydirection tag contains a list of three doubles defining the up direction vector for the text string presenting the measurement value.

Content model

A list of three doubles.

Attributes

None.

textsize

The textsize tag contains the size for the text string presenting the measurement value. Note that the text associated with 3D measurements and comments scales with the view; so the text size is defined as the size in inches when the view is first activated.

Content model

Double.

Attributes

None.

markupcolor

The markupcolor tag contains a list of three doubles defining the RGB color used in presenting the measurement markup.

Content model

A list of three doubles.

Attributes

None.

value

The value tag defines the numerical value representing a measurement value. This value is converted to a text string and displayed as part of the measurement text string.

Content model

Double.

Attributes

None.

units

The units tag contains a text string indicating the units of the associated measure value. This is appended to the value string when presenting the measurement.

Content model

Text String.

Attributes

None.

precision

The precision tag defines the number of decimal digits shown for the measurement value. If not specified, the default is three decimal places.

Content model

Integer.

Attributes

None.

usertext

The usertext tag contains an optional text string that is appended to the end of the measurement.

Content model

Text String.

Attributes

None.

extensionlength

The extensionlength tag defines the length of the extension line in points. The default value is 60 points.

Content model

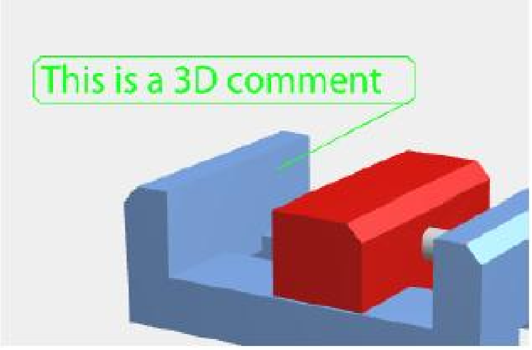
Double.

Attributes

None.

3D Comment Related Elements

3D comment notes allow users to connect a comment to a specific piece of geometry in the 3D model. The markup consists of a leader line that connects the model to a text box placed in the 3D scene. The text box is rendered so that the text is always facing the user. The anchor point defines the connection to the model. The model space text position tag defines placement of the corner of the text box that lies closest to anchor point 1 in the current view. The user string field contains the text that is to be fitted into the text box. If the text box is not large enough, the string is truncated.



The following is an example of a 3D comment.

```
<measure>
 <measurename>3DComment1</measurename>
 <comment3dmarkup>
      <anchor>0.008982 -0.019600 0.039761</anchor>
      <anchorpartname>Base</anchorpartname>
      <textboxx>13</textboxx>
      <textboxy>1</textboxy>
      <textposition>0.067426 -0.086996 0.089997</textposition>
      <textsize>20.000000</textsize>
      <markupcolor>0.000000 0.996078 0.000000</markupcolor>
      <usertext>This is a 3D comment</usertext>
 </comment3dmarkup>
</measure>
```

comment3dmarkup

The comment3dmarkup tag is a container that holds the data defining an instance of a 3D comment associated to a geometric item.

Content model

`(anchor & anchorpartname? & textposition & textsize? & textboxx & textboxy & markupcolor? & usertext?)`

Attributes

None.

anchor

The anchor tag contains a list of three doubles defining the model space position of the first anchor point in world space. It is assumed that this is a position on the 3D model associated with this view.

Content model

A list of three doubles.

Attributes

None.

anchorpartname

The anchorpartname tag defines the node name for the part that the point anchor lies on.

Content model

Text String.

Attributes

None.

textposition

The textposition tag contains a list of three doubles defining the model space position of the text anchor (the start of the text string) associated with the measurement.

Content model

A list of three doubles.

Attributes

None.

textsize

The textsize tag contains the size for the text string presenting the measurement value. Note that the text associated with 3D measurements and comments scales with the view; so the text size is defined as the size in inches when the view is first activated.

Content model

Double.

Attributes

None.

markupcolor

The markupcolor tag contains a list of three doubles defining the RGB color used in presenting the measurement markup.

Content model

A list of three doubles.

Attributes

None.

textboxx

The textboxx tag defines the x size of the character box containing the comment. The tag contains the number of character columns within the box.

Content model

Integer.

Attributes

None.

textboxy

The textboxy tag defines the y size of the character box containing the comment. The tag contains the number of character rows within the box.

Content model

Integer.

Attributes

None.

usertext

The usertext tag contains the text string that is shown in the comment box.

Content model

Text String.

Attributes

None.