

Tagged PDF Best Practice Guide

PDF Association, PDF/UA Competence Center | Document Version 0.1.1 | January 19, 2015

Table of Contents

| | |
|-------------------------------------------------------------------------|----|
| Document History | 2 |
| Background | 2 |
| About PDF/UA | 2 |
| Introduction to the Structure Elements Best Practice Guide | 2 |
| What this Guide is not..... | 2 |
| Additional reading | 2 |
| A view towards PDF 2.0..... | 3 |
| PDF documents | 3 |
| PDF pages | 3 |
| Artifacts | 3 |
| Content that spans pages..... | 3 |
| Page-centric editing processes..... | 3 |
| Standard structure types in ISO 32000-1 (PDF 1.7) | 4 |
| Grouping elements..... | 4 |
| Block level structure elements..... | 11 |
| Inline elements..... | 21 |
| Illustration elements | 33 |
| Empty structure elements..... | 39 |
| Attributes and properties | 40 |
| List attributes | 40 |
| PrintField attributes | 40 |
| Table attributes | 40 |
| Commonly-used properties of content..... | 41 |
| Text characteristics | 47 |
| Superscripts and Subscripts..... | 47 |
| Symbolic characters | 47 |
| Other features of PDF | 47 |
| Digital Signatures | 47 |

Document History

| Version | Date | Change |
|---------|------------|---------------|
| 1.0 | 2016-xx-xx | First edition |

Background

PDF is the world's chosen electronic document file format. The value of PDF derives from consistency and stability in presentation to end users.

Some basic realities of consuming PDF files are essential for implementations that seek to provide a consistent experience of accessible PDF files.

- AT must present structure elements in a suitable way.
- Just as AT informs the user when a <Figure> is encountered, AT should be capable of inform the user of, for example <Reference> tags . The manner in which this information is communicated is not prescribed.

About PDF/UA

Ensuring content is accessible to users with disabilities presents broad and complex challenges in any technology. ISO 14289-1 (PDF/UA-1) specifies technical requirements for PDF files to ensure a high-quality and consistent reading experience when used by a variety of PDF/UA-conforming processors.

Introduction to the Structure Elements Best Practice Guide

This document is intended for developers implementing tagged PDF and PDF/UA.

Others who may benefit from using this Best Practice Guide include those performing detailed accessibility testing on PDF files claiming conformance with PDF/UA, or claiming to be accessible according to some other specification.

What this Guide is not

This Guide does not provide step-by-step guidance for achieving PDF/UA conformance, nor does it offer information specific to any particular software application.

Additional reading

[AIIM \(www.aiim.org/Research-and-Publications/standards/committees/PDFUA/\)](http://www.aiim.org/Research-and-Publications/standards/committees/PDFUA/)

- PDF/UA-1 Technical Implementation Guide: Understanding ISO 14289-1 (PDF/UA-1)
- PDF/UA-1 Technical Implementation Guide: Understanding ISO 32000-1 (PDF 1.7)
- Achieving WCAG 2.0 with PDF/UA

[PDF Association \(www.pdfa.org\)](http://www.pdfa.org/)

- The Matterhorn Protocol
- PDF/UA in a Nutshell

A view towards PDF 2.0

ISO 32000-2 (PDF 2.0), expected to be published in 2017, makes many changes to tagged PDF. However, basic principles will be unchanged. Implementations that follow the guidance offered in this document will find that investment entirely transferable to PDF 2.0, and the eventual PDF/UA-2, when they are published.

PDF documents

(metadata, etc)

PDF pages

Artifacts

This document provides no additional guidance regarding Artifacts beyond the provisions of PDF/UA and ISO 32000.

Content that spans pages

Logical structure is agnostic to pagination. This implies that content items that span two (or more) pages will have to be linked to the logical structure twice in the right order without restarting the content item.

Example: a single paragraph starts on page 4 and continues on page 5. From the logical structure point of view this paragraph is a single content item, but encoded in two parts on the page description level.

Page-centric editing processes

Deleting pages, splitting documents, inserting pages and similar operations require that not only the page objects are handled appropriately, but also the tagging structures (as well as bookmarks or internal link destinations) connected with these content objects.

Example: When deleting pages, parts of the tagging structure connected to the deleted pages should also be removed.

Example: When inserting pages into a tagged PDF document, care must be taken to reconcile the tagging structure of the inserted pages with that of the target document.

Standard structure types in ISO 32000-1 (PDF 1.7)

Grouping elements

Part, Art, Sect, Div

ISO 32000-1 does not provide clear guidance on appropriate use of these structure elements, and nor does PDF/UA-1.

Creation Rules

As a general guideline implementers may want to adhere to the following:

<Part>, <Art> and <Sect> should be used for semantic grouping. While ISO 32000-1 seems to imply nested use of <Part>, <Art> and <Sect>, implementers may use them as they see fit.

<Div> should be used for non-semantic grouping, such as associating an attribute (such as language) with a sequence of block-level elements, or for role-mapping a custom grouping structure element for which none of the other grouping elements are suitable.

Consuming Rules

Implementations should be able to reflect the semantic grouping indicated by <Part>, <Art> and <Sect> elements.

Quote and BlockQuote

Encloses quoted content.

Example:

<xx>

Creation Rules

Where quoted content exists inside a paragraph or other block-level structure element, the <Quote> tag is used.

Where quoted content does not exist inside a paragraph or other block-level structure element, the <BlockQuote> tag is used. Example: for longer portions of quoted content.

Consuming Rules

Quoted content should be presented such that a consumer can distinguish between quoted and unquoted content.

For example, TTS could use voicing changes or beeps to indicate a quote, whereas a visual consumer using text extraction / reflow may benefit from a text styling change.

Caption

In PDF 1.7 <Caption> is described as follows:

- A brief portion of text describing a table or figure
- A <L> may contain a <Caption> as its first element (prior to the actual items)
- A <Table> may include a <Caption> as its first or last child element
- A <TOC>. See <L>, above

Although not strictly forbidden in ISO 32000-1 or PDF/UA, it is not best-practice to include content as the direct child of an structure element.

ISO 32000-1 14.8.4.2 Table 333's definitions should be interpreted as more general than the expressly-stated cases. For example, certain situations imply that other content elements also require <Caption> in order to correctly represent the content's semantics. This issue was addressed in ISO 32000-2 as follows:

ISO/DIS 32000-2 states:

For lists and tables a Caption structure element may be used as defined for the L (list) and Table structure elements. In addition a Caption may be used for a structure element or several structure elements in the following way:

- Structure elements are understood to be “captioned,” or assigned a caption when a Caption structure element exists as an immediate child of the same parent structure element.
- The Caption should be the first or the last structure element inside the parent structure element.
- If a Caption structure element does not occur as the first or last structure element inside a grouping structure element it may be treated in the same way as a Div or P structure element.

While captions are often used with figures or formulas, they may be associated with any type of content.

When used for a Table or L (list) structure element, a Caption may be present as a structure element inside the Table or L (list) structure element, or outside the Table or L (list) structure element but inside the list's or table's parent structure element, or both. The presence of both uses of the Caption structure element for the same Table or L (list) structure element should be avoided. If nevertheless a Caption structure element is present for both, the outer caption must be associated with the combination of inner caption and the Table or L (list) structure element respectively.

Creation considerations

Sidebar content offers challenges for implementers of tagged PDF in PDF 1.7, and especially, those sidebars that include content appearing to include headings.

The use of heading tags within sidebars is incorrect as it results in breaking the document's logical structure. As a work-around for PDF 1.7, in such cases, and as they are semantically more appropriate than <P> elements for this use case, <Caption> structure elements may be used to enclose the apparent headings within the sidebar.

Users may be advised that a work-around is to linearize the structure entirely, eliminating the problem, but at the cost of being forced to include the sidebar content within the overall document's logical structure.

Implementers should note that in PDF 2.0 the <Aside> and <Document> structure elements provide a solution to tagging sidebar content.

TOC / TOCI

Although structurally, <TOC> and <TOCI> tags are very similar to L and LI tags, <TOC> / <TOCI> tags differ from conventional lists (<L> and structure types) because they offer pointers into the document rather than offering distinct content.

These elements should be used not only for Table of Contents but also for Tables of Figures or Illustrations, etc.

PDF 2.0 Advisory: Although <TOC> and <TOCI> structure elements are not standard structure elements in PDF 2.0, these structure types are elements of the PDF 1.7 tagset, which is the default tagset of PDF 2.0.

Use of TOC tags for multilevel tables of contents

<TOC>

<TOCI>

<TOC>

<TOCI>

Links

Links are not required in TOC/TOCI structures in PDF 1.7 or PDF/UA-1.

PDF 1.7 does not identify the <Link> element as one that may be present within a <TOCI>, however, <Link> elements may exist within <TOCI> elements.

<Reference>

Content

<P>

A special instance...

Leaders

Dot leaders should be marked as artifact. Do not use the <NonStruct> element, as described with respect to TOC/TOCI, (ISO 32000-1 Table 333).

Creation Rules

XX

Consuming Rules

XX

Index

Essentially, a special-purpose <Div> element used with Indexes. It may replace <Div> for the purpose of grouping <Reference> nodes.

Example:

```
<Index>
  <P>
    <Reference>
    <Reference>
  <P>
    <Reference>
```

or

```
<Index>
  <L>
    <LI>
      <Reference>
      <Reference>
    <LI>
      <Reference>
```

Creation Rules

In principle, any structure element may be used inside a <Index> element.

Typically, <Index> elements are organized as lists, and thus, in such cases, <L> and would be used.

Consuming Rules

Must offer an optional mechanism to indicate the fact of an <Index> and to provide Indexes as available targets of navigation (distinct from headings).

Nonstruct

Has no role or meaning; interpretation is out of scope for consumers of Tagged PDF, and may be considered a utility for private purposes.

Example:

None needed.

Creation Rules

This element is useful only for private purposes.

Consuming Rules

Ignore it.

Private

Akin to NonStruct, this element differs from NonStruct insofar as not only the structure element, but the content of the element are also ignored.

Example:

None needed.

Creation Rules

This element is useful only for private purposes.

Consuming Rules

Ignore the element and its contents.

Block level structure elements

P

A generic block level set of content that is not otherwise specified with another block level structure element type such as heading, table or list elements.

Example:

<P>

Content

Creation Rules

As content may not be placed directly inside grouping structure elements, always write a <P> tag inside a grouping element unless the content is a heading, table or list.

<P> is used to separate paragraphs. It is not acceptable to enclose several paragraphs with a single <P> tag, or to directly nest <P> tags.

Inline elements are welcome inside <P> tags. While in principle any structure element may have a value of “inline” as a placement attribute, and thus, may be a child of a <P> tag, this practice is generally reserved for block level structure elements.

While empty <P> tags are not explicitly prohibited they should be avoided.

Consuming Rules

Some viewers may wish to reflow text enclosed in <P> tags. In such cases a visible separation of the contents of individual <P> tags is conventional.

H1-H6

In PDF, headings and subheadings are the means of semantic organization of content in a document. The H1-H6 tags identify the headings from which the logical structure of the document can be derived.

The fact alone that some text is formatted using bold face or large type, or is intended to be very important, is not a sufficient reason to use a heading tag.

Despite the fact that PDF/UA-1 requires headings not be skipped, otherwise well-structured documents exist in which headings are skipped and where modification of the document is not an option.

It is not acceptable to split a single heading into two or more structure elements themselves tagged as headings.

Correct example:

```
<H2>Talking about titles</H2>
```

Incorrect example:

```
<H2>Talking about</H2>
```

```
<H2>titles</H2>
```

NOTE: Line breaks are not a concept that can be expressed using tagged PDF.

A subheading should not be enclosed within a heading tag; in most cases, subheadings are enclosed in <P> tags. The same applies to other content found next to headings such as tag-lines.

Headings are not to be confused with document titles, for which no structure type exists in ISO 32000-1.

Talking about titles

A title is information representing the normal means of referring to the document. Titles can be present as both metadata entry and page-content.

- The metadata entry for a PDF document's title is represented using XMP.
- A document's title appearing as page content traditionally has often been tagged with <H1>.

Since PDF/UA does not require any specific structure type for title content, it is permissible to tag such content with either <H1> or other tags (typically, <P> or tags mapped to <P>).

Since headings are commonly appear in Tables of Contents, and since document titles do not normally appear in Tables of Contents, it can be argued that it is preferred to use the <P> tag, and not the <H1> tag, for tagging title content.

Page content representing the title can appear several times in the document. If <H1> tags are used to enclose such content, only one such instance of the title should be tagged as <H1>.

NOTE: PDF 2.0 adds a <SE_Title> structure type, and so this guidance will change substantially for PDF 2.0 and PDF/UA-2. This implies that a document prepared for conformance with PDF/UA-1 will be difficult to convert to PDF/UA-2 if <H1> is used for the title. If the <SE_Title> structure type encloses the title content as a custom tag, role-mapped to <P>, then conversion to PDF/UA-2 can be achieved by simply removing the role-mapping.

Examples

Example A

```
<H1> (first heading, no title present)
```

```
<P>
```

<H2>

<P>

Example B

<P> (encloses document title)

<H1>

<P>

<H2>

<P>

Example C

<SE_Title> (encloses document title)

<H1>

<P>

<H2>

<P>

Example D

<H1> (encloses document title)

<H2>

<P>

<H2>

<P>

In examples A, B and C, it is acceptable to use the <H1> headings several times. In example D it is not acceptable, as no content can be on the same nesting level as the document title.

Creation Rules

Since ISO 32000-1 has no structure type matching the concept of title, such content should not be tagged with <H1> or <H>. However tagged, such content's enclosing tag should be mapped to <P>.

Tools creating PDF/UA documents may insert empty heading tags to fill the gap that would otherwise be left by skipped heading levels.

Consuming Rules

A common use of headings is to dynamically generate a table of contents (or other navigational mechanism).

Be prepared to encounter empty heading tags.

H (strongly structured)

Due to a lack of suitable tools, this tag is impractical, and its use is not recommended.

Example:

None offered.

Creation Rules

None offered.

Consuming Rules

None offered.

Lbl

<Lbl> encloses content that labels other content.

[TBD]

L (List), LI, LBody

The <L> (list) tag encloses a sequence of one or more content items. The tag is used to enclose each content item. <Lbl> encloses the list item marker for each content item. <LBody> is used to enclose the content item's semantic content.

For PDF 1.7 (and thus, PDF/UA-1), if a caption is present it's enclosed in a <Caption> tag and precedes the tags.

The Caption standard structure type has been substantially overhauled in PDF 2.0.

Example of a simple list:

```
<L>
  <LI>
  <LI>
    <Lbl>
    <LBody>
  <P>
```

Multilevel lists

Although a canonical form can be derived from ISO 32000-1, there is no mandatory provision that prohibits the use of other forms of parent-child relationships between the various standard structure elements for lists.

As a result, processors should be able to handle various forms of lists, and also, lists in which list-item content encapsulates its own list, not semantically associated with the overall enclosing list.

Some examples of acceptable list structures are offered below.

Example of a canonical multilevel list

```
<L>
  <LI>
    <Lbl>
    <LBody>
  <LI>
    <Lbl>
    <LBody>
    <L>
      <LI>
        <Lbl>
        <LBody>
      <LI>
        <Lbl>
        <LBody>
    <LI>
      <Lbl>
      <LBody>
```


Another “legal” form of nested list

This model (multilevel list) is borrowed from HTML.

```
<L>                                // <ol> or <ul> in HTML
    <LI>                           // <li> in HTML
        <L>                        // <ol> or <ul> in HTML
```

Another “legal” form of nested list

```
<L>
    <LI>
    <L>
```

Example of a list containing yet another list

In this example the two <L> structure elements represent independent structures.

```
<L>
    <LI>
        <Lbl>
        <LBody>
    <LI>
        <Lbl>
        <LBody>
            <P>
            <P>
            <L>
                <LI>
                    <Lbl>
                    <LBody>
                <LI>
                    <Lbl>
                    <LBody>
            <P>
    <LI>
        <Lbl>
        <LBody>
```

Creation Rules

<L> tags would typically include an optional <Caption> tag and one or more tags.

For each content item that has a list item marker – such as a bullet or list-numbering – that list item marker must be enclosed in a <Lbl> tag. If a content item does not have a list item marker the <Lbl> tag must not be present. It is not allowed to enclose list item markers within <LBody> tags or directly within tags.

List item markers may be of any content type, including images, graphical shapes or text. All the rules of tagged PDF apply to such items, including the use of Alt and ActualText attributes, as well as requirements for text including mapping to Unicode.

The entire semantic content of each content item is enclosed in an <LBody> tag. Depending on the nature of the semantic content, inline content or arbitrary complex structures may be enclosed within the <LBody> tag. It is not permitted to place semantic content directly under the tag.

Table, TR, TH, TD, THead, TBody, TFoot

Authoring tools may follow these rules to produce reliably accessible table structures in PDF. For example, well-formed tables using colspan or rowspan in TD cells may be automatically rendered as complex tables when converted to PDF.

Example (Simple Table)

```
<Table>
  <TBody>
    <TR>
      <TH>
      <TH>
    <TR>
      <TD>
      <TD>
```

Creation Rules

General

- Tables that span multiple pages must be tagged as a single table.
- Empty cells must be TD.
- Rows or columns of TH cells may include individual TD cells.
- If there is a semantic reason to use a Complex table then do so; if there is no semantic reason, use a Simple table.
- For rows that contain totaling or other information that summarizes or groups rows or columns it's best to use the simplest solution (ie, an approach that allows one to use a Simple vs. a Complex table per these guidelines). As a practical matter this means avoiding the use of rowspan and colspan.
- Cells used for presentation only are not permissible. An example would be an empty row or column separating content that should be represented as two (or more) tables.
- Empty cells are acceptable (unless used for presentation purposes alone, as noted above).

Simple Tables

Simple tables rely on a rectangular tabular structure in which all data cells may be uniquely identified by either row-headers, column headers, or both.

- TH cells must sit at the top or left (in LRTB writing systems).
- The Scope attribute is required on all TH cells.

Complex Tables

Complex tables use header and ID attributes to associate TD cells with their respective TH cells. A table is complex when:

- TD cells use colspan or rowspan attributes
- A TH does not apply to all cells in a row or column

Consuming Rules

Use case: Header-finding algorithm from PDF 2.0?

Use case: Highlight based on the <Div>

Use case: "Where am I" feature, for example, to identify the current cell.

Inline elements

Span

The primary purpose of is to demarcate content for the purpose of applying semantic, presentational or other attributes. Although the element itself has no inherent semantics, semantics come from attributes applied to it.

 may be a vehicle for presentational, language (Lang), replacement content (ActualText) and alternate description (Alt) and expansion (E) attributes.

Example:

```
<P>  
  This is the  
  <Span E='National Aeronautics and Space Administration'>  
    NASA  
  </Span>  
  press release.  
</P>
```

Creation Rules

None

Consuming Rules

None

Quote

ISO 32000-1, Table 338 defines the <quote> tag as: “An inline portion of text attributed to someone other than the author of the surrounding text.”

Example:

<P>

ISO 32000-1, Table 338 defines the <Quote> tag as:

<Quote>

“An inline portion of text attributed to someone other than the author of the surrounding text.”

</Quote>

</P>

Creation Rules

None

Consuming Rules

When voicing text, consider changing voice.

Note

Note tags encompassing footnotes and endnotes operate in conjunction with Reference tags.

Example 1 (Note without reference)

```
<p>
<Note Placement=Block>
<p> (from next page)
```

Example 2 (An inline footnote)

A possible method for handling <Reference> and <Note> tags is as follows:

```
<p>
    <Reference>
        <Lbl>
    <Note>
        <Lbl>
```

The following extended example includes a representation of a page-rendering followed by a representation of the structure appearing on the page.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

The first reference¹ for the first footnote on this page, the second reference² for the second footnote on this page, and so on.

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

```
<P>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.</P>
```

```
<P>The first idea<Reference><Lbl>1</Lbl></Reference> <Note><Lbl>1</Lbl>) The first idea is to understand footnotes</Note> for the first footnote on this page, the second idea<Reference><Lbl>2</Lbl></Reference> <Note><Lbl>2</Lbl>) for the second idea is to make them work in PDF</Note> for the second footnote on this page, and so on.
```

```
<P>Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.</P>
```

Example 3 (Commonly-encountered work-around)

WARNING: This approach is not official, but well-known and otherwise high-quality agents use this work-around to the vagueness in ISO 32000. This circumstance should be processable.

Simply put, developers should ideally be prepared to process instances in which <Note> tag(s) are inserted immediately after the structure element that contains the <Reference> tag(s) for these Note(s).

In the following extended example the logical presentation (in the structure tree) of the footnotes is shown highlighted while the physical rendering remains at the bottom of the page. As in Example 2, above, a representation of the structure follows.


Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

The first reference¹ for the first footnote on this page, the second reference² for the second footnote on this page, and so on.

1) The first idea is to understand footnotes

2) The second idea is to make them work in PDF

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.



1) The first idea is to understand footnotes

<P>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.</P>

<P>The first idea<Reference><Lbl>¹</Lbl>/Reference> for the first footnote on this page, the second idea<Reference><Lbl>²</Lbl></Reference> for the second footnote on this page, and so on.</P>

<Note><Lbl>1</Lbl>) The first idea is to understand footnotes</Note>

<Note><Lbl>2</Lbl>) The second idea is to make them work in PDF</Note>

<P>Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.</P>

Example 4 (Footnote with link)

Footnotes and endnotes may occur without links; links may (of course) occur with content other than footnotes or endnotes.

```
<p>
    <link> -- Destination
        <Reference>
            <Lbl>
... ..
    <Note> (the Destination of the Link)
        <Lbl>
        <P>
```


Example 5 (Footnotes / endnotes referenced from multiple locations)

Although ISO 32000-1 does not describe an explicit mechanism to address this use-case, the use-case is readily resolved by taking advantage of the mirroring of the footnote / endnote <Lbl> inside the <Reference> tag.

Creation Rules

<Note> should only be used for footnotes and/or endnotes.

As of PDF 1.7 association between the <Reference> and its <Note> tag may most readily be accomplished by the sequence in the logical order.

A Link on the Reference may target a Destination (ISO 32000-1, 12.3.2 “Named Destinations”), however, use of this approach requires detailed session navigation in AT. Absent such navigation users to return to the <Reference> tag in the text (this applies to all document-internal links; not just in the Reference/Note context).

Semantics of supplemental or explanatory content “notes”

The term “note” in running text is commonly used to identify supplementary content for content contained within another element. Such “notes” should not be tagged using the <Note> tag. Instead, such cases are distinguished by means of a grouping element, as in the following example:

| | |
|-------|--------------------------------|
| <Div> | A grouping element |
| <P> | Content |
| <P> | A note pertaining to the table |

Especially in the context of <Figure>, <Formula> or <Table> structure elements, <Caption> may be the appropriate tag.

General

Notes may be inline, block or grouping elements, and therefore may include substructures.

PDF 2.0 Considerations

PDF 2.0 makes it possible to explicitly associate References with Notes via the **Ref** key in the structure element dictionary.

Consuming Rules

- AT must provide functionality that:
 - Presents <Reference> and <Note> tags in a suitable way
 - Can inform the user when a <Reference> is encountered
 - Provides navigation to the associated <Note> content
 - Allows a user to return to the <Reference> after visiting the matching <Note>
 - Skips <Note> tags when reading text sequentially
- AT should take into account that <Note> tags can be ILSE, BLSE or grouping elements, depending on usage.

NOTE 1: AT developers should be aware that <Reference> and <Note> tags do not exist in HTML.

NOTE 2: Due to limitations in the underlying PDF specification, although ID attributes on <Note> tags are required by PDF/UA, they currently (PDF 1.7) provide no added value to consuming processors.

Optionally, provide a mechanism not to present <Note> content unless visited from a <Reference>.

Reference

Reference tags encompass content that refers to other content, typically, <Note> structure elements.

Example (Reference)

```
<p>  
<Reference>
```

Creation Rules

Whether a <Reference> tag is used in the context of a cross-reference or in the context of a footnote / endnote may be distinguished by the presence of a <Lbl> tag:

- For a <Reference> tag containing a <Lbl> tag it is assumed to point to a footnote / endnote. The contents of the <Lbl> tag in both cases should exactly match. No footnote / endnote present between a <Reference> tag and the corresponding footnote / endnote must have the same <Lbl>.
- For a Reference tag not containing a <Lbl> tag it is assumed to be a cross-reference. This also applies to <Reference> tags contained in <TOCI> tags as defined in ISO 32000-1 Table 333.

PDF 2.0 Considerations

Although PDF 2.0 defines a new tag set that does not include a <Reference> standard structure element, PDF 2.0 readers supporting Tagged PDF are required to support the PDF 1.7 tag set as the default namespace.

In PDF 2.0, the <Reference> concept is provided by the **Ref** key (known in the early 2014 CD of PDF 2.0 as the **Destination** key) in the structure element dictionary (PDF 2.0, 14.7.2).

By combining the <Reference> structure element with the **Ref** key introduced in PDF 2.0 it is possible to create hybrid elements that work in both PDF 1.7 and PDF 2.0.

Consuming Rules

2015-03-19: In Adobe's PDF Accessibility API, <Reference> structure elements should be of Type StructElement, not of Type Link.

Rationale:

In the case of nested <Reference> and <Link> tags, which is a common circumstance, the current Type of Link for <Reference> elements causes (understandably) consuming AT to repeat link advisories. The Type of Link is properly reserved for Link tags (which must include Link annotations).

Based on the creation rules it is always possible to determine the corresponding <Note> for a given footnote / endnote <Reference>.

When encountering a footnote / endnote <Reference> structure element, use the contents of the <Lbl> tag as an identifier. Search forward for the first <Note> structure element containing the identifier as the content of its <Lbl> structure element.

It is possible to find <Reference> from a given <Note>'s <Lbl> by means of scanning backwards in the document to find the matching <Lbl> in a <Reference>.

NOTE: AT developers should be aware that <Reference> and <Note> tags do not exist in HTML.

An AT wishing to take advantage of <Reference> can offer a means to navigate from a <Reference> tag to the matching <Note> tag... and back. Such implementations should take into account that more than one <Reference> may point to the same <Note> tag.

BibEntry

Intended to semantically identify individual entries in a bibliography, this structure element simply serves to group content for reuse.

Example:

None.

Creation Rules

None offered.

Consuming Rules

None offered.

Code

Intended to semantically identify code examples, this structure element serves to indicate that enclosed content should be represented precisely; without further modification (for example: justification, cleanup of white space).

Use of this structure element does not imply that extraction would result in usable code.

Example:

None.

Creation Rules

None offered.

Consuming Rules

None offered.

Link

The <Link> structure element encloses linked content, if any, and actionable link annotation(s). Actual linked content is not necessary to the use of this structure element.

Example:

```
<P>
    Visit the
    <Link>
        website
        OBJR-xxx (http://...)
    </Link>
    today!
</P>
```

Creation Rules

Without QuadPoints, some cases (text breaking across several lines is a very common example) may lead to multiple annotations. Support for QuadPoints allows for more reliable experiences with interactive viewers.

To avoid ambiguity, link annotations enclosed by a single <Link> structure element must have the same action.

Links that span pages

Although in general OBJRs may appear at any location within the <Link> element, for links that span pages, all OBJRs should be next to each other in the logical order.

EXAMPLE

```
PAGE 7
    <Link>
        Text 1
        OBJR 1
PAGE 8
        OBJR 2
        Text 2
    </Link>
```

Consuming Rules

The user may prefer the provision of an option for choosing how to announce multiple link annotations inside one <Link> element. Options may (but are not restricted) include:

- In the case where a single <Link> structure element encloses multiple link annotations all appearing on the same page, and where all link annotations are the same; it may be ideal to represent a single annotation to the user.

- In the case where a single <Link> structure element encloses multiple link annotations that appear on more than one page, and where all link annotations are the same; it may be useful to represent those annotations that appear on a single page as a single annotation to the user.

Annot

The <Annot> structure element encloses annotations other than links and widgets.

There are two classes of annotations:

- Markup annotations, or annotations used like markup annotations. The <Annot> structure element encloses the marked-up content and the object reference to the actual annotation. These may be nested.
- Other annotations. The <Annot> structure element typically only encloses the reference to the actual annotation.

Example:

```
<Annot>  
  Content  
  OBJR (pointing to <Annot> of type Highlight)
```

Creation Rules

A challenge can arise from the fact that content to be marked up by a markup annotation is not already represented as a structure element, and cannot be directly associated with the annotation. Instead, it might be necessary to identify and isolate the marked-up content and enclose it in marked-content sequences which can then be associated with the <Annot> structure element together with the actual annotation.

Some housekeeping might be necessary regarding other marked-content sequences and structure elements around the marked-up content.

Consuming Rules

When presenting marked-up content to a user, the following aspects should be included in that presentation:

- The marked-up content
- The fact that the content is marked-up, and the type of markup annotation
- The contents of the Annotation's content entry
- Any annotations that are replies to the Annotation

While not mandated by conforming reader provisions in PDF/UA-1, processors should also make available annotation properties, for example: author, subject, status, date/time, checkmark.

Ruby and Warichu (et. al)

No guidance provided at this time.

Illustration elements

Figure

A grouping element is a “Figure Grouper”, and should be presented as a logical unit if:

- If at least one <figure> tag is enclosed within the element.

NOTE: In this context the Document tag is not considered a grouping element.

Example: Figure with Caption

| | |
|-----------|----------------------------------------------------|
| <Div> | The Figure Grouper grouping tag |
| <Caption> | The Figure’s caption |
| <Figure> | The tag enclosing the actual image or illustration |

Example: Figure without Caption

While it’s preferred for Figures to have captions there are valid use-cases where a figure would not have a caption (e.g.: a logo, title-page image, in-line graphics such as smileys, symbols or other illustrative content).

<Figure>

Creation Rules

Any Grouping tag may be used (Div, Part, Sect, etc.) as the “Figure Grouper” in the following manner:

NOTE: “<Div>” is used in these rules for brevity.

- Each <Div> must include one Caption, and no more than one Caption
- Each <Div> may include more than one Figure
- A “Figure Grouper” <Div> may nest within another such <Div>

Enclosing all Figures with appropriate Figure Grouper tags (<Div> or otherwise) including those without captions, or without related content, is recommended.

Consuming Rules

- When encountering a <Div>, check for a enclosed <Caption> tag. When encountered, consider the <Div> as described by the caption, including enclosed Figure tags and other elements.
- Content enclosed with a Figure Grouper <Div> should be presented as a logical unit.
- It may be useful to provide a facility for highlighting content enclosed within a <Div>.

Formula

Encloses content normally perceived as a formula or equation, whether used inline or as a display formula (block level). The use of <Formula> is not limited to math, but may (in principle) be applied in other areas of science such as chemistry or physics.

Individual symbols may or may not be enclosed in a <formula> based on context.

Example:

<Formula>

$$2 + 2 = 4$$

Creation Rules

For formulas represented with raster and/or vector images, do not use a <Figure> tag since these objects are not, semantically, figures.

Consider using MathML as custom tags (PDF 1.7), map them to . For table-like structures, consider leveraging table tags. Map the math tag to <Formula>.

A Formula tag must include an Alt attribute (PDF/UA-1, 7.7).

Content inside a Formula tag may make use of an ActualText attribute, but should only include an ActualText attribute on very small pieces of content. The ActualText attribute should be on the respective tag, not the Formula tag.

Form

Form fields in ISO 32000 come in various flavors. Some types of form fields have one interactive object (a widget annotation) associated with them (pushbutton or text form field), others have (possibly) several interactive objects (several widget annotations) associated with them (radio button groups).

Each <Form> element encloses a single widget annotation.

Contrary to what one expects, a <Form> element encloses a single widget annotation, not a form field. This implies that for form fields with several widget annotations, several <Form> elements are necessary. There is no pre-defined element in the logical structure of the document that ties the widget annotations belonging to the same form field together. Nonetheless, a grouping element such as <Div> may be used for this purpose.

ISO 32000 does not provide any specific mechanism for identifying form field labels. Such association can only be provided by way of context and logical ordering (i.e., placement of the label next to the form field and/or the <Form> elements associated with it).

Although not strictly required under all circumstances, the most straightforward approach to providing an alternate description for form fields is to provide the TU entry in the form field dictionary, which is typically presented when the user has focus on the widget annotation.

For a discussion about non-interactive forms, see the section on PrintField attributes.

Example

PDF forms require both logical and Acroform structures in alignment to produce desired results.

A simple radio-button example:



Anrede: ☒ Frau ☐ Herr

| Logical Structure | Acroform Structure |
|-----------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> <H2> "Anrede:" <P> <Form> widget "Frau" <Form> widget "Herr" </pre> | <pre> << /FT/Btn /Ff 33603584 % - "radio button" /T (Anrede) /TU (Anrede) /V /Frau /Kids [<< /Type /Annot /Subtype /Widget /AS /Frau /AP << /N << /Off << ... >> % - Appearance of "off" state /Frau << ... >> % - Appearance of "Frau" state >> >> ... >> << /Type /Annot /Subtype /Widget /AS /Off /AP << /N << /Off << ... >> % - Appearance of "off" state /Herr << ... >> % - Appearance of "Herr" state >> >> ... >>] >> </pre> |

Core concepts in radio button form fields

Linking logical structure to radio button form fields

A radio button form field contains widgets for each button within the radio button group. The <Form> structure element in the logical structure always points to widgets. For a form field with two radio buttons, two <Form> structure elements are required to link them to the logical structure. There is no explicit link between the logical structure and the form field. Furthermore, there is no explicit link

between text in the page’s description describing a radio button widget or the form field enclosing it and <Form> structure elements in the logical structure.

How to find out the meaning of each radio button

Each widget in a radio button form field should contain two entries in the appearance dictionary’s N (normal) entry. One of these entries should have the name “Off”; the other entry has a name that represents the meaning of that radio button (e.g. “Frau” and “Herr” in the above example).

How to get the current value of the current radio button form field

The current value of a radio button form field is represented by the V entry in the radio button form field. In the case of missing V entry the value is “Off”.

The Opt entry

The optional Opt entry accommodates cases where the value of each radio button is not suitable for presentation to human users. Using the Opt array, each radio button’s value can be associated with a more human-readable version of that value. The example shown below uses “F” and “H” for the values; the Opt key provides more readable versions of these values in the form of “Frau” and “Herr”.

| Logical Structure | Acroform Structure |
|-----------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> <H2> "Anrede:" <P> <Form> widget "Frau" <Form> widget "Herr" </pre> | <pre> << /FT/Btn /Ff 33603584 % - "radio button" /T (Anrede) /TU (Anrede) /V /F /Opt [% - order must match that of the /Kids array (Frau) (Herr)] /Kids [<< /Type /Annot /Subtype /Widget /AS /F /AP << /N << /Off << ... >> % - Appearance of "off" state /F << ... >> % - Appearance of "F" state >> >> ... >> << /Type /Annot </pre> |

| | |
|--|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | <pre> /Subtype /Widget /AS /Off /AP << /N << /Off << ... >> % - Appearance of "off" state /H << ... >> % - Appearance of "H" state >> >> ... >>] >> </pre> |
|--|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Creation Rules

Form field labels in page content will in many cases be the same as the content of the TU key.

When “flattening” interactive form fields, use PrintField attributes to ensure the (now) non-interactive form is represented in an accessible fashion.

Consuming Rules

Interactive viewers encountering forms using PrintField attributes should indicate that such forms are “read only”.

Readers should avoid an approach in which the user has only form fields presented, and by implication, may miss other content.

Empty structure elements

Structure elements may be “empty” in that they do not enclose content. It should be possible to represent such structure elements to the user, even if such representation is not the default.

Attributes and properties

List attributes

ListNumbering

The possible values for this attribute are limited with respect to numbering system (to upper and lower alpha and upper and lower Roman). Arbitrary labels use the value *None*.

The specification does not address the concept of continued lists (list fragments with other block level elements between fragments); in fact, this construct is simply unavailable in 32000-1.

Reuse in HTML

Processors intending representation in HTML should consider that a **ListNumbering** attribute value of *None* implies that contents of the <Lbl> tags in the list should be used to represent the list's labels.

Example

```
<Lbl>Day 1:</Lbl>  
<Lbl>Day 2:</Lbl>
```

PrintField attributes

With respect to PrintField attributes, radio buttons and list boxes are not properly accommodated in ISO 32000-1. These attributes are entirely overhauled in PDF 2.0.

Table attributes

For ordinary tables the **Scope** attribute is generally the easiest approach.

For complex table structures, including subdivisions within tables, Headers and IDs generally have to be used in order to correctly represent the table's structure.

Where **Scope** is sufficient, right arrow = *Row*, down arrow = *Column*

| ↓ | ↓ | ↓ |
|---|---|---|
| ■ | | |
| ■ | | |
| ■ | | |

Headers entry in each **TH** cell has to enumerate the **IDs** of table cells (could be **TDs** and **THs**!)

If on a **TH** the **Headers** entry implies that the **TH** itself again points to some other table cells, which allows for expression of nested table headers hierarchies. The use of Headers in a table implies that the enumerated cells have to have an **ID** attribute.

| | ↓ | ↓ |
|---|---|---|
| ■ | | |

| | | |
|---|---|---|
| ■ | | |
| ■ | | |
| | ↓ | ↓ |
| ■ | | |
| ■ | | |
| ■ | | |

Summary attribute. Use of this attribute should be restricted to cases where visual information about the table would not be characteristically available to assistive technology.

Where auxiliary information or guidance would be useful to any user it should be provided in text, and not hidden in a Summary attribute which would only be available to those using certain AT.

Providing a Summary is not precluded for specific target audiences, but the practice should be limited to such cases.

Commonly-used properties of content

PDF provides for a rich relationship between properties of content. These properties may be combined to accommodate a variety of use-cases.

The four keys in question can appear in both the property list dictionary (for marked content) and the structure element dictionary (for structure elements).

- **Lang** (language) defines the natural language of both the content and the values of the **E**, **Alt** and **ActualText** properties present in the same context.
- **E** (expansion of abbreviations and acronyms) provides additional information representing expanded text for both the values of the **Alt** and **ActualText** properties present in the same context.
- **Alt** (alternate description) provides descriptive information for content with a substantial non-textual aspect.
- **ActualText** (replacement text) is the text representation of text that is not encoded as text.

Alt (alternate description)

Provides descriptive information for content with a substantial non-textual aspect. The use of this property depends on the visual appearance of the content; it is not a function of the data object type used. For example, ASCII art consists of characters and yet, being a visual representation, nonetheless requires an **Alt** property.

On the other hand, a piece of text may be encoded as an image object. As such the appropriate property for accessibility purposes would be **ActualText** repeating the text represented by the image, and the **Alt** property is inappropriate.

In cases where the fact of an image is relevant to the user (for example, a scan of a small section of a historical document), it may be appropriate to use both **Alt** and **ActualText** properties.

While most commonly used on the <Figure> tag, **Alt** needs to be used for any content that is mostly not text-based, including cases where that content consists of a number of distinct graphics objects.

Examples include:

- Pie charts

- Technical drawings and flow charts
- Clip-art
- Maps

Nested structures

Illustrations may include subdivisions that can be considered as illustrations in their own right. In such cases an **Alt** property is appropriate for both the main illustration as well as those contained within it. Examples include:

- Diagrams of sub-assemblies
- Washing instructions
- Maps including countries and states

Example

No syntax example is provided.

For examples of best practice in writing Alt values, see.... <to be provided>

Creation Rules

As applied to Tagged PDF, although **Alt** is technically usable in both marked content and structure element contexts there are few (if any) use cases in which **Alt** would be appropriate on marked content.

ActualText

Applied either on the marked content or structure element level, this attribute is useful in a variety of circumstances, including:

- A figure that represents text
- In certain cases hyphenation causes characters to change (example: Drucker -> Druk-ker)
- As a last resort only, when other approaches are not feasible:
 - To specify text directly (for example, to insert white space to separate words)
 - To specify the absence of text (for example, ensuring that a soft hyphen to break a word between lines, is not presented)
 - When deciding between applying ActualText in marked content vs. a structure element

Example

[TBD]

Creation Rules

For the purposes of this clause, “text content” is content that is normally (visually) perceived as text regardless of whether that content is encoded as a text object, an image object, a vector object, a clipping path, a mask or any combination thereof.

Scanned pages

The use of either **Alt** or **ActualText** on a scanned page is almost always inappropriate. In many cases, scanned pages are overlaid with invisible text where the text matches the position of the scanned text and can be tagged according to the rules of PDF/UA.

In the case of semantic images or other graphical objects on the scanned page they have to be tagged as such using the <Figure> tag. This will necessitate some means of selective inclusion of that portion of the page in the page description.

Known limitation(s)

In the case of <Figure>, <TD> or <TH> structure elements, some popular APIs recognize the fact of an ActualText attribute by replacing the <Figure> element with that attribute's value.

SPECULATION: Some ATs examine the type (StructElem, Text) of a node; if a StructElem is encountered, AT handles it as a structure element.

Work-around: elements may include the ActualText attribute on a or similar nonstructural element contained within the semantic element.

Although cases exist (for example, remediation) in which it may be easier to implement ActualText on a structure element, it's frequently preferable to address ActualText via marked content.

If it isn't possible to provide a ToUnicode table, ActualText is also useful for representing Unicode for characters that do not naturally map to Unicode (for example, bullets, or when associating a ligature with a sequence of separate characters).

Whenever possible, software should provide text content as text objects in the page description (this includes white-space characters). If not possible or if difficult, use (in the following order):

1. ActualText on the content, or
2. ActualText on an empty structure element. This approach should only be used for white-space characters.

When the content is already contained within a structure element, and is the complete content of that element, then it is preferred to place ActualText on the structure element.

NOTE: For content exceeding short sequences of characters that cannot be encoded as text objects in the page description (such as a scanned and OCR'd page), the text is best superimposed as text objects in render mode 3 (invisible text).

NOTE: Use of ActualText is limited to text that would otherwise be contained within a single structure element.

Consuming Rules

[TBD] – see earlier (2015-10-22) discussion.

How should AT work?

TBD: Three choices...

1. **Backwards-compatible with current behavior:** Require elements with ActualText to be nested within the semantically appropriate tag, or
2. Change the PDF spec (obviously, in PDF 2.0) to make it clear that ActualText does NOT “replace” the structure element, or
3. Require consuming software to ignore the structure element type in the case of <Figure> tags alone.

E

Expansion text is used for abbreviations and acronyms. Its use is not required by PDF/UA.

Example



Creation Rules



Consuming Rules

Any viewer should be able to indicate the presence of an E attribute, and display the contents to users.

Lang

A language identifier is a language code whose specificity (e.g., *fr* vs. *fr-ca*) may vary. Non-linguistic text content gets the *zxx* language code, which declares the absence of linguistic information.

Lang exists at four levels in a PDF document:

- Document level: a **Lang** entry in the Catalog
- Logical structure level: a **Lang** attribute of a logical structure element
- Content level: a **Lang** property of a marked content sequence
- Text level: a Unicode escape sequence

A **Lang** entry in the Catalog establishes the default language for the entire document, including page content, metadata and text strings within annotations.

A **Lang** attribute of a logical structure element or a marked content sequence can be used to override the default. It applies to all child elements unless overridden by a logical structure element or marked content sequence nested inside. Logical structure and marked content may be nested in any fashion.

A Unicode escape sequence can be used to set the language inside any text string including entries in annotations and **Alt**, **ActualText** and **E** (expansion) entries. Support for Unicode escape sequences is currently not available in existing PDF processors.

For some XMP metadata fields of type **Lang Alt** (Language Alternate) it is possible to include language-specific instances in addition to a default instance.

Example

ISO 32000-1 provides adequate examples.

Consuming Rules

Lang governs all attributes of the structure element on which it appears, and all enclosed content.

Text characteristics

Superscripts and Subscripts

For text-to-speech, text customization and text repurposing applications, the superscript or subscript aspect of characters may be critical to distinguishing the contents of a Reference's label to the surrounding text, and for other purposes.

There's no explicit mechanism in Tagged PDF that can express superscript or subscript. Although the BaselineShift attribute may be used as a substitute, it cannot be considered a reliable mechanism, as text could be shifted for other reasons.

NOTE: PDF 2.0 includes the standard layout attribute "TextPosition".

Symbolic characters

Where symbolic characters appear as text, use Unicode to the extent possible, either by means of ToUnicode table entries, or by means of ActualText, where the ActualText would contain the appropriate Unicode. Where no Unicode code point is available, use appropriate text conveying the meaning within an ActualText attribute.

Where symbolic characters appear as, or like figures, use the <Figure> structure element.

Consuming software should take advantage of Unicode encoding, and be prepared to present Unicode code points beyond the predominant script or language used. This includes, for example, mathematical symbols, Zapf-Dingbats, WingDings.

Other features of PDF

Digital Signatures

Digital signatures in ISO 32000-1 use a few conventions that do not lend themselves well to the tagged PDF paradigm. Accordingly, the PDF/UA requirements for digital signatures may trigger a few questions.

For example: digital signatures are customarily not clearly "visible" or "invisible" – they are often placed in a signature form field made functionally invisible by virtue of encoding with /Rect [0 0 0 0]. It's hard to know how such signatures are to be represented, or indicated in logical structure, or indeed, whether they need to be located in the logical structure. Here we offer some general principles for handling such cases.

Reading order of digital signatures

Fields that are of zero size, or outside the CropBox, or are hidden are considered "invisible" and thus do not have to be included in the tags tree as is otherwise required for Annotations in 7.18.

However, Consistent with clause 8.6 of PDF/UA-1, conforming readers are required to provide reasonable access to digital signatures irrespective of their visibility.

The most appropriate way of representing invisible signatures to users with disabilities will generally be via a separate user interface, not by artificially forcing the digital signature into the logical reading order.

Requirements for field appearances

Since objects within a digital signature appearance stream cannot be tagged, 100% compliance with 7.13 is impossible in cases where such appearance streams include logical substructure (such as a

<Figure>). In many cases an appropriate Alt entry on the Form tag for the digital signature should be sufficient.