



Tagged PDF Best Practice Guide: **Syntax**

For developers implementing ISO 14289-1 (PDF/UA)

Version 1.0.1 (January 2023)

Copyright © 2023 PDF Association

This work is licensed under the Creative Commons Attribution 4.0 International License.

To view a copy of this license, visit <http://creativecommons.org/licenses/by/4.0/>
or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

PDF Association
Neue Kantstrasse 14
14057 Berlin, Germany

Tel: +49 (0)30 39 40 50-0
Fax: +49 (0)30 39 40 50-99

E-mail: copyright@pdfa.org
Web: www.pdfa.org

Published in Germany and the United States of America

Tagged PDF Best Practice Guide: **Syntax**

Table of Contents

1	Background.....	4
1.1	Use of the term “tagged PDF”	4
1.2	Document history.....	4
2	Introduction.....	5
2.1	About PDF/UA (Universal Accessibility).....	5
2.2	Accessibility vs. reuse.....	5
2.3	What this Guide is not	5
2.4	Syntax guidance vs. tagging guidance	5
2.5	Looking towards PDF 2.0 and PDF/UA-2	5
2.6	Use of normative language	6
2.7	Notation.....	6
3	General provisions.....	7
3.1	Scope	7
3.2	Fundamentals	7
3.3	Document level attributes	8
3.4	Content that spans pages	9
3.5	Empty structure elements	9
3.6	Role maps	9
3.7	Artifacts.....	10
4	Guidance for the standard structure types.....	11
4.1	Grouping elements.....	11
4.2	Block level structure element types	21
4.3	Illustration elements.....	47
5	Attributes and properties.....	54
5.1	Layout attributes.....	54
5.2	List attributes	58
5.3	PrintField attributes	59
5.4	Table attributes.....	59

5.5	Commonly-used properties of content	60
6	Text characteristics.....	66
6.1	Superscripts and subscripts	66
6.2	Symbolic characters.....	66
7	Other features of PDF	66
7.1	Digital signatures	66
7.2	Page open options	67
8	Editing tagged PDF files.....	67
	Annex A: The PDF/UA flag.....	68
	Annex B: PDF 2.0.....	69
A.1	Differences between PDF 1.7 and PDF 2.0.....	69
A.2	Namespaces and standard structure types	69
A.3	Investing in PDF 2.0 while supporting PDF 1.7	69
	Bibliography	71

1 Background

This Best Practice Guide is intended to help developers understand the syntactical characteristics of tagged PDF files required for an accessible experience. Other Best Practice Guides will address other aspects of accessible PDF content.

1.1 Use of the term “tagged PDF”

“Tagged PDF” is the title of 14.8, the clause defining PDF’s accessibility mechanism in ISO 32000-1. This term may be somewhat confusing, as ISO 32000 uses the term “structure element” instead. A “tagged” PDF is one which meets the requirements of clause 14.8 in ISO 32000, which includes structure element types, structure attributes and word disambiguation.

This document uses the term “tagged PDF” to refer to PDF documents that include structures enabling accessibility. It uses the more technically-correct term “structure element” to refer to the <P>, <Figure> and other container objects that are known to most users as “tags”.

1.2 Document history

Version	Date	Change
1.0.1	2021-01-10	Minor changes
1.0	2019-06-07	Version 1.0 released
0.1	2015-12-22	Initial pre-release

2 Introduction

This document is intended for developers implementing tagged PDF and PDF/UA. Others (including authors with some technical knowledge of PDF's accessibility mechanisms) may also benefit from this document. For example, this Guide is intended to be useful for those performing detailed accessibility testing on PDF documents claiming conformance with PDF/UA, or on PDF documents claiming to be accessible according to some other specification.

2.1 About PDF/UA (Universal Accessibility)

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 1.7 (ISO 32000-1) -conforming files to ensure a high-quality and consistent reading experience when used by a variety of PDF/UA-conforming processors and assistive technology.

2.2 Accessibility vs. reuse

Tagged PDF facilitates content reuse for many purposes; accessibility is one such purpose. Although tagging for certain reuse purposes might differ from tagging for accessibility purposes, in general, best practice for accessibility purposes is also best practice for typical reuse objectives.

The focus of this guide is the establishment of the **best possible use of PDF 1.7 syntax for accessibility** in the form of guidance on translating common content organization and semantics into structure elements.

2.3 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. This Guide does not provide information on syntax validation against PDF 1.7 and does not claim to provide complete coverage of the use of structure elements.

Although this document may be useful to technically-inclined authors, it is not intended as a guide for document authors seeking information on how to tag PDF files. It does not address WCAG 2.1 or Section 508, but it can be used to inform activities intended to achieve conformance with such requirements.

2.4 Syntax guidance vs. tagging guidance

This Syntax guide does not address all issues of interest to developers, in particular, guidance on selecting tags in various circumstances.

Typical creation software simply transfers the user's selection of structuring element (however chosen) to the corresponding output irrespective of actual semantic appropriateness. PDF/UA-aware software might offer advice if an incorrect or questionable usage occurs.

Guidance in semantically-correct tagging will be addressed in other PDF Association publications.

2.5 Looking towards PDF 2.0 and PDF/UA-2

Although ISO 32000-2 (PDF 2.0) makes many changes to tagged PDF, the basic principles are unchanged. See Annex B: PDF 2.0 for more information. PDF/UA-2 (ISO 14289-2) will be based

upon PDF 2.0. It is generally consistent with PDF/UA-1, but there are important differences, including newly-defined and revised structure elements and clarifications on nesting limitations.

2.6 Use of normative language

In ISO standards, these terms are considered “normative”, in that they have specific, defined meanings:

- “shall” = required (to avoid confusion with the specifications, this term is not used in this Guide)
- “should” = strongly recommended
- “may” = permitted

This Best Practice Guide cannot and does not substitute for PDF/UA-1 itself, and thus only uses ISO normative terms when quoting ISO or other third-party specifications.

2.7 Notation

The following conventions are used throughout:

- In the document’s text, key names are given in **boldface**
- In example pseudo-code, standard structure element entries (e.g., for examples) are given with angled-brackets (e.g., <Div>). The elements are not closed; instead, items contained within structure elements are enclosed by “{ }”. Attributes are indicated using HTML conventions, e.g. ‘<P lang=“en-us”>’, remarks or special characters are shown by [].

Example:

```
<Figure alt=“PDF icon”>
<Caption> {
    <P> [remark or notice]
    <P> {relevant content}
}
```

3 General provisions

3.1 Scope

The guidelines in this clause are generally applicable to all content within a PDF/UA-1 document.

3.2 Fundamentals

The most fundamental requirement for conformance with PDF/UA-1 is introduced clause 7.1, paragraph 2:

“Content shall be marked in the structure tree
with semantically appropriate tags in a logical reading order.”

This clause discusses the significance and application of PDF/UA’s requirement.

3.2.1 Semantic appropriateness

From time to time, this document will use the term “semantically appropriate” to indicate a reference to ISO 14289-1, 7.1, paragraph 2.

Ensuring correct semantics for structure element types makes it possible for assistive technology users to understand roles and relationships in the document. PDF/UA requires conforming software to generate tagged PDF in which the tags fully represent the author’s semantic rather than literal intent.

Examples demonstrating the requirements of PDF/UA-1, 7.1, paragraph 2 as applied to use cases:

- Heading content must be tagged with the correct structure element (<H1>–<H6>), to provide an unambiguous relationship between the heading content and other content in the document
- It is semantically inappropriate to include list bullets inside <LBody> structure elements; semantic appropriateness requires that such content be tagged with <Lbl> structure elements within the same as the pertaining <LBody> element.
- If Header and ID attributes are not provided, and do not adequately identify the header relationship, each table header cell (<TH>) requires the Scope attribute.
- References for footnotes must be tagged to reflect all the appropriate semantics, e.g. (<Reference><Lbl>1</Lbl></Reference>)
- Soft line-breaks (typically added by end users with Shift + Return) do not indicate creation of a new BLSE.

Example:

Input in the application:

```
{Talking about [line break] titles}
```

PDF output:

```
<H2> {Talking about titles}
```

NOTE Line breaks are not a concept that can be expressed using PDF 1.7.

In terms of correct use of semantic structures in word-processing applications, it’s common for authors to make mistakes, e.g.:

- A table was used to format some content, but semantically, the content is simply a set of paragraphs.
- Tabular information is not structured with table structure elements
- A heading is separated in two headings because the author added a line-break for layout purposes

3.2.2 Reading order

A fundamental requirement for accessibility is the ability to unambiguously understand the logical order and sequencing of text and other objects. In PDF, this information is often unavailable because PDF files are traditionally created with print and display as the primary goal.

A logical reading-order established via a depth-first traversal of the structure tree is a critical feature of accessible PDF files.¹

3.2.3 Mapping text to Unicode

Text (page content, metadata, annotations) must be mapped to Unicode in order to be accessible (see ISO 32000-1, 14.8.2.4.2, “Unicode mapping in Tagged PDF”).

In cases in which Unicode mappings are not available (e.g, a logo encoded as text), mapping to the Unicode private use area (PUA) is the only solution that facilitates conformance to encoding requirements in PDF/UA-1. However, this results in a loss of semantic information. In order to retain such information, **Alt** or **ActualText** properties can be used (see 5.5, “Commonly-used properties of content”).

3.3 Document level attributes

PDF/UA-1 regulates document level attributes as follows:

- The document level XMP metadata must include the PDF/UA “flag” (see Annex A).
- The document level XMP metadata must contain a Title (dc:title) (see PDF/UA-1, 8.11, “Metadata”).
- The document’s View property must be set to display the Title (not the file name) (see PDF/UA-1, 8.12.1 “Metadata”).

PDF/UA-1 does not explicitly require specification of a document’s language at the document level (via the **Lang** entry in the Catalog dictionary), but simply requires that all content have a declared language. Accordingly, instead of the **Lang** entry in the document’s catalog dictionary, the document’s content could be enclosed in structure elements or marked content sequences that have the correct **Lang** attribute. However, PDF/UA-1 implicitly requires a document-level language in order to indicate applicable language for metadata and outline entries on the document level. as otherwise the applicable language for the Title entry, or any other metadata fields or entries in the outline entries (commonly referred to as bookmarks) would not be known.

¹ Reading order is further addressed in the forthcoming Tagged PDF Best Practice Guide: Tagging.

3.4 Content that spans pages

Logical structure is agnostic to pagination. Accordingly, content items that span two (or more) pages must each be linked to the logical structure, in the right order, without restarting the structure element.

Example: a paragraph starts at the bottom of page 4 and continues at the top of page 5. From a logical structure perspective, this paragraph is a single content item, and is enclosed by a single `<P>` structure element. However, from the page description level, the same paragraph is encoded in two parts, each part a marked content sequence linked to the very same `<P>` structure element.

3.5 Empty structure elements

In PDF 1.7, structure elements may be “empty” in that they do not enclose content directly or indirectly. However, it is semantically appropriate that an empty structure element only be present if there is a semantic reason for its inclusion (e.g., an empty `<TD>` structure element within a table to ensure correct table structure). In general, and to minimize the chances for confusion, it is recommended that structure elements not be empty unless they serve a semantic role in a defined substructure such as a `<Table>` or `<L>` (list).

It is important for consumer software to be able to represent empty structure elements to the user (e.g. an empty cell in a table still needs to be conveyed for correct semantics and table understanding).

Certain structure element types (e.g., `<Figure>` or `<BlockQuote>` elements) preclude (by nature of their semantics) usage without content.

It is semantically acceptable for the following types to be empty:

Structure element type	Example (non-exhaustive) use cases
TD	Maintain table structure
LI	Maintain list structure
Span	ActualText for white-space characters; metadata properties; attributes
Div	Provide metadata properties and/or attributes
Document	A single-page document with no content
NonStruct and Private	Inclusion of arbitrary tagsets

Empty heading structure elements (`H#`, `H`) are discussed in 4.2.2, “`<H1>—<H6>` (headings)”.

It is semantically inappropriate for all other structure element types to be empty. However, empty structure elements are a reality in real-world documents whether permitted by PDF/UA or not. The ability to handle such cases is advised.

3.6 Role maps

Tagged PDF defines a set of standard structure types (see ISO 32000-1:2008, 14.8.4) to enable interchange of document semantics, but PDF creators are not limited to this base set, and may

extend it through the use of custom structure types. In such cases, **RoleMap** entries (see ISO 32000-1, Table 322) are required that map these custom structure types (e.g., “<DataTable>”) to semantically-appropriate standard structure types (e.g., <Table>). Accordingly, it would be incorrect to map <DataTable> to <H1>.

3.7 Artifacts

The process of laying out and paginating content for display can lead to the introduction of additional display items (e.g. page numbers on each page or table borders). These items are not part of what ISO 32000-1 defines as “real content”; they are considered artifacts of layout (see 14.8.2.2, “Real Content and Artifacts” in ISO 32000-1). A requirement for tagged PDF is to clearly distinguish “real” content from artifacts. PDF/UA also makes it clear that artifacts must be accessible, but it is less specific about precisely what is required for content marked as Artifact.

Artifact content must be accessible, therefore the basic rules of accessibility (see 3.2 “Fundamentals”) apply, including requirements for reading order and Unicode.

It is semantically inappropriate to contain semantic content within a marked content sequence tagged as artifact.

3.7.1 Header and footer content

Page headers and footers are usually placed automatically as a function of pagination. As such this content is not part of the reading-order of the document and is not considered to be “real content.

3.7.2 Page numbers

Page numbers must be marked as artifacts in marked-content sequences with a property list entry **Pagination** (see ISO 32000-1, Table 330 – Property list entries for artifacts) property).

Accessible page enumeration is enabled through use of the Page Labels (ISO 32000-1, 12.4.2). It is semantically appropriate to have Page Label values match the visible page number.

4 Guidance for the standard structure types

4.1 Grouping elements

4.1.1 <Part>, <Art>, <Sect>, <Div>

<Part> is intended to sub-divide a large document into smaller elements. <Art> identifies an article within a document or part. <Sect> identifies the sections of a document, part or article. <Div> is a division of a document without semantic intent.

Nether ISO 32000-1 nor PDF/UA provides detailed guidance on semantically appropriate use of these structure element types.

4.1.1.1 Example

Example A:

```
<Part> [e.g. category of a magazine] {  
    <Art> [one article in a category] {  
        <H1> [main content of the article]  
        <P>  
        <Sect> [info box content] {  
            <P>  
            <Div lang="de-DE"> [passage of foreign language content,  
where a Lang attribute is assigned to <Div>] {  
                <P>  
                <P>  
            }  
        }  
    }  
    <Art>  
    ...  
}
```

4.1.1.2 Creation

See example above.

4.1.1.3 Consumption

To adequately enable users to navigate larger documents it is strongly recommended that implementations be able to reflect the semantic grouping indicated by <Part>, <Art> and <Sect> elements.

4.1.2 <BlockQuote>

<BlockQuote> encloses longer portions of quoted content and can be used as a block-level element or a grouping element, whereas <Quote> encloses quoted content within a paragraph (inline element).

4.1.2.1 Examples

Example A: (a block-level quote used as a block level element)

```
<P>  
<BlockQuote> {content}  
<P>
```

Example B: (a block-level quote – used as a grouping element – including substructure)

```
<P>  
<BlockQuote> {  
    <P> {content}  
    <P> {content}  
}  
<P>
```

4.1.2.2 Creation

No specific guidance provided.

4.1.2.3 Consumption

It is recommended that quoted content be presented such that a consumer can distinguish between quoted and unquoted content. For example, text-to-speech (TTS) could use voicing changes or beeps to indicate a quote, whereas a visual presentation using text extraction / reflow may benefit from a text styling change.

4.1.3 <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

Where not otherwise defined for uses other than <Table>, <L> and <TOC>, it is recommended that <Caption> structure elements be contained within a parent structure element that also contains the content being captioned. A caption is expected to occur directly above or below the item it captions. For <Figure> structure elements, for which current-generation AT does not expect child structure elements, <Caption> elements are expected to occur immediately before or after as siblings of the captioned element (see “Example A”).

For elements that allow the explicit inclusion of a <Caption> as a child element (e.g. <Table>), the <Caption> must be the first or last direct child (see “Example C”).

4.1.3.1 Examples

Example A: (<Caption> including substructure, i.e. a single caption including two paragraphs)

```
<Figure>
<Caption> {
    <P>
    <P>
}
```

NOTE: In practice, most implementations accept direct content within a <Caption>.

Example B: (<Caption> for a table where the <Caption> occurs logically prior to the table itself)

Examples B and C are both equally valid; the choice between them is typically made on the basis of visual presentation order in the PDF, or on the basis of reuse considerations.

```
<Table> {
    <Caption> [substructure, e.g. <P>]
    <TR>
    <TR>
}
```

Example C: (<Caption> for a table where the <Caption> occurs logically following the table itself)

```
<Table> {
    <TR>
    <TR>
    <Caption> [substructure, e.g. <P>]
}
```

Example D: <Caption> for a list

```
<L> {  
    <Caption> [substructure, e.g. <P>]  
    <LI>  
    <LI>  
}
```

4.1.3.2 *Creation*

In PDF 1.7 <Caption> is only intended for use with <Figure>, <Table>, <List> and structure element types, the definitions in ISO 32000-1 14.8.4.2.

4.1.3.3 *Consumption*

Processors encountering a <Caption> immediately preceding or following a <Figure>, <L>, <Table> or <Formula> structure element are recommended to assume that the <Caption> refers to that structure element, i.e. as if the <Caption> structure element were inside that <Figure>, <L>, <Table> or <Formula> structure element.

Processors are recommended to be prepared to encounter <Caption> elements associated with <Formula> and <TOC> structure element types.

4.1.4 <TOC> (table of contents) / <TOCI> (Table of contents item)

Although <TOC> and <TOCI> structure element types are very similar to <L> and structure element types in structural terms, table of contents structure element types differ from conventional lists (<L> and structure element types) because they explicitly provide references into the document rather than semantically distinct content.

It is recommended that <TOC> / <TOCI> structure element types be used for all types of tables of contents, including tables of figures or illustrations, etc. There is no restriction on the number of tables of contents within a document.

4.1.4.1 Examples

Example A: (top-level structures usage of <TOC> & <TOCI> in a multilevel table of contents)

```
<TOC> {  
  <TOCI> {Chapter 1}  
  <TOCI> {Chapter 2}  
  <TOC> {  
    <TOCI> {Chapter 2.1}  
    <TOCI> {Chapter 2.2}  
  }  
}
```

Example B: (substructures in a <TOC> & <TOCI> context containing <Reference> without a link)

```
<TOC> {  
  <TOCI> {  
    <P> {  
      <Reference> {  
        <Lbl> [In cases where the TOCI is numbered] {  
          1.  
        }  
        <Span> {  
          Introduction ..... page 5  
        }  
      }  
    }  
  }  
}
```

If the table of contents uses link annotations it is recommended to use <Link> structure elements within <Reference> elements.

Example C: (substructures in a <TOC> & <TOCI> context containing <Reference> & <Link>)

```
<TOC> {  
    <TOCI> {  
        <P> {  
            <Reference> {  
                <Link> {  
                    <Link-OBJR>  
                    <Lbl> [In cases where the TOCI is numbered] {  
                        1.  
                    }  
                    <Span> {  
                        Introduction ..... page 5  
                    }  
                }  
            }  
        }  
    }  
}
```

4.1.4.2 Links

PDF 1.7 does not permit a <Link> element as a direct child of a <TOCI>, however, <Link> elements commonly do exist (even though they are not required or suggested by ISO 32000-1 or PDF/UA-1) within <TOCI> elements.

4.1.4.3 Creation

Where a <TOCI> contains text beyond the reference to the respective chapter or other section of the document, the <TOCI> may contain one or more <P> structure elements to enclose both the reference and the additional text. Since ISO 32000-1 prohibits inline elements from containing block-level elements, <Link> elements within a <TOCI> are recommended to have the **Placement** attribute set with a value of *Block*.

PDF 1.7 implies that the <NonStruct> structure element type or Artifact marker for marked content sequences can be used to enclose dot leaders. Accordingly, it is recommended to avoid the <NonStruct> element in this case, and simply mark dot leaders as Artifact instead.

4.1.4.4 Consumption

Irrespective of the definitions in PDF 1.7, it is recommended that processors expect to encounter <Link> structure elements within <TOCI> structure elements.

It is recommended that processors expect to encounter documents containing multiple Tables of Contents.

4.1.5 <Index>

<Index> is a grouping element for document indices. If present, it has a descriptive element and a reference element (i.e., referring into the body of the document).

4.1.5.1 Examples

Example A: (index with a nested list as contents, giving more information about relationships)

```
<Index> {
  <L> [index as a list] {
    <LI> [index topic] {
      <Lbl> [section identifier, e.g. "B"]
    <L> [list containing entries relating the topic "B"] {
      <LI> {
        <Lbl> [first entry, e.g. "Beer"]
        <LBody> [containing all references] {
          <Reference> [first page number, e.g. 20]
          <Reference> [second page number, e.g, 22-24]
          <Reference> [see also Food]
        }
      <LI> {
        <Lbl> [numbering, if available, e.g. "Boy"]
        <LBody> [containing all references]
          <Reference> [first page number, e.g. 28]
          <Reference> [second page number, e.g, 29]
          <Reference> [see also Girl]
        }
      <LI> [index topic] {
        <Lbl> [section identifier, e.g. "C"]
      <L> [list containing entries relating the topic "C"] {
        <LI> {...}
      }
    }
  }
}
```

Example B: (without alphabetical structure)

```
<Index> {
  <L> [index as a list] {
    <LI> [index topic] {
      <Lbl> [section identifier, e.g. "appetizer"]
    <L> [list containing entries relating the topic "appetizer"] {
      <LI> {
        <Lbl> [first entry, e.g. "bruschetta"]
        <LBody> [containing all references] {
```

```

        <Reference> [first page number, e.g. 20]
        <Reference> [second page number, e.g. 22]
    <LI> {
        <Lbl> [numbering, if available, e.g. "Caesar
Salad"]
        <LBody> [containing all references]
        <Reference> [first page number, e.g. 28]
    <LI> [index topic] {
        <Lbl> [section identifier, e.g. "main dishes"]
        <L> [list containing entries relating the topic "main dishes"]
    {
        <LI> {...}
    }
    }
}

```

4.1.5.2 *Creation*

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

Typically, <Index> elements are organized as lists, and thus, in such cases, <L> and would be used. <P> is often used, too. Although PDF/UA-1 does not forbid it, to avoid confusion with the main body of the document it is recommended to avoid the use of heading elements inside <Index> elements.

Note that an index is often preceded by a heading (e.g. "Index").

4.1.5.3 *Consumption*

It is recommended that consuming software offer an optional mechanism to indicate the fact that an <Index> is present, and to provide such an Index as an available target of navigation (distinct from headings).

4.1.6 <NonStruct>

Has no substantive role or meaning; interpretation is out of scope for consumers of tagged PDF. NonStruct's value is primarily as a utility for role mapping custom structure element types for which no corresponding standard structure element type is suitable.

4.1.6.1 *Examples*

None provided.

4.1.6.2 *Creation*

A document containing custom structure elements for which there is no corresponding standard structure type, but where the content and child elements are intended to be real content. In such a case appropriate semantics require a writer to role map the custom element type to NonStruct.

4.1.6.3 *Consumption*

NonStruct has no semantic significance, but its content and contained structure elements may well have significance. A consuming processor would be expected to pass these children to AT and other processors.

4.1.7 <Private>

Akin to <NonStruct>, this element differs from <NonStruct> insofar as not only the structure element itself, but the children of the structure element, including structure elements and content, are also ignored.

4.1.7.1 *Examples*

None provided.

4.1.7.2 *Creation*

This element is useful only for private purposes.

4.1.7.3 *Consumption*

Ignore the element and its contents.

4.2 Block level structure element types

4.2.1 <P> (paragraph)

Paragraphs comprise the most common content type in most documents. <P> generally encloses distinct portions of content that are not otherwise specified with other block level structure element types such as heading, table or list elements.

<P> is often a good “backup” choice when no other structure type is semantically appropriate, or as a fallback in role-mapping.

4.2.1.1 *Example*

Example A

```
<P>    {content of the paragraph}
```

4.2.1.2 *Creation*

As ISO 32000-1 prohibits content as a direct child of grouping structure element types (see ISO 32000-1, 14.8.4.2 “Grouping elements”), in the absence of other semantics, it is typically appropriate to write a <P> structure element inside a grouping element unless the content is a heading, table or list.

<P> is used to identify individual paragraphs. It is not semantically appropriate to enclose several paragraphs with a single <P> structure element, or to directly nest <P> structure elements.

4.2.1.3 *Consumption*

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

4.2.2 <H1>—<H6> (headings)

PDF/UA-1 requires heading-levels not be skipped (e.g., <H1>, <H2>, <H4>). However, otherwise well-structured documents exist in which heading-levels are skipped and where modification of the content is not an option. In such cases, although a PDF/UA flag cannot be used, it is recommended that the file to conform with PDF/UA in all other respects.

PDF/UA-1 requires the use of <H7> and higher heading levels if semantically appropriate. Such structure types are undefined in PDF 1.7. PDF/UA requires a mapping for undefined elements; accordingly, in this case it is recommended that <H7> and higher levels of headings be role mapped to <P> or <H6>, depending on the nature of the content and the heading structure in it. If it is more important to not mis-represent a heading of a nesting level higher than <H6> as <H6>, mapping to <P> is recommended, whereas if it is more important to represent a heading of a higher nesting level than H6 as a heading, rather than turning it into a paragraph, mapping to <H6> is recommended.

There is no structure element type or syntax for subheadings. Instead, use, <P> and/or structure types.

4.2.2.1 Examples

Example A: (in a single paragraph)

```
<H1>  {The Mothers [e.g. huge, bold typeface] [line break]
      <Span> {Fillmore East - June 1971 [e.g. small, regular typeface]
      }
}
```

Example B: (in two paragraphs)

```
<H1>  {The Mothers}
<P>    {Fillmore East - June 1971}
```

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

4.2.2.2 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. In PDF:

- The metadata entry for a PDF document's title is represented using XMP.
- A document title appearing as page content is commonly tagged with <H1>.²

Since PDF/UA-1 does not require any specific structure type for title content, it is permissible to structure such content with either <H1> or other structure element types (typically, <P> or structure element types mapped to <P>).

² The background of this unfortunate reality is complex; its discussion is out of scope for this document.

Page content representing the title can - especially in publications - appear several times in the document. If <H1> structure elements are used to enclose such content, it is recommended that only one such instance of the title be structured as <H1>.

Since headings commonly appear in tables of contents, and since document titles do not normally appear in tables of contents, a future-proof (PDF 2.0) approach would be to use a <Title> structure type (which is defined in PDF 2.0, see Annex B) mapped to the <P> structure type. Upgrading this document to PDF 2.0, therefore, would then simply require deletion of this role map.

4.2.2.3 Sidebars

Sidebar content offers challenges for implementers of tagged PDF in PDF 1.7, and especially, those sidebars that include content styled in a manner similar to that of headings in the main body of content.

Since PDF 1.7 does not include a structure element type specific to sidebars or similar content, when heading structure elements are used in a sidebar, they must be semantically appropriate in the context of the document as a whole, that is, be consistent with the other heading structure elements in the document.

As a work-around for cases in which sidebars include content that would (if tagged with heading structure elements) not be semantically appropriate in the context of the overall document, then <P> structure elements are likely to be the most semantically appropriate structure type for the “headings” within the sidebar.

Another workaround is to use a <Div> element with a layout attribute appropriate to the content. Although many AT implementations do not yet support such a construct, such <Div> elements can provide processors with a means of distinguishing such content.

Another work-around is to linearize the structure entirely, eliminating the problem, but at the cost of being forced to include the sidebar’s content within the overall document’s logical structure.

4.2.2.4 Examples

Example A:

```
<H1> [first heading, no title present]
<P>
<H2>
<P>
```

Example B:

```
<P> [encloses document title]
<H1> [first main heading, e.g. “1. Top level heading”]
<P>
<H2> [heading, 2nd level, e.g. “1.1 2nd level heading”]
<P>
```

Example C:

```
<Title> [encloses document title (role-mapped to <P>)]
<H1> [first main heading]
```

<P>
<H2>
<P>

Example D:

<H1> [encloses document title]
<H2> [first main heading]
<P>
<H2>
<P>

In examples A, B and C, it is semantically acceptable to use the <H1> headings several times. Example D, a case of <H1> used for a title (which is not a violation of PDF/UA-1), is only semantically appropriate if there is only one <H1> in the document.

4.2.2.5 *Creation*

Even though ISO 32000-1 has no structure type matching the concept of “title”, it is recommended that such content not be structured with <H1> or <H> unless it is the only <H1> in the document (as per Example D, above). If the creator chooses to create a <Title> (or similarly-intended) structure element type (which is acceptable), it is recommended to map such structure element types to <P>.

WARNING: Tools creating PDF/UA documents may insert empty heading structure element types to fill the gap that would otherwise be left by skipped heading levels. This behavior cannot result in conformance with PDF/UA or WCAG 2.x.

4.2.2.6 *Consumption*

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

Since heading levels above 6 are explicitly permitted in PDF/UA, it is recommended that processors be prepared to encounter heading levels above 6 irrespective of mapping.

4.2.3 <H> (heading, strongly structured)

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

4.2.3.1 *Examples*

None offered.

4.2.3.2 *Creation*

No specific guidance provided.

4.2.3.3 *Consumption*

No specific guidance provided.

4.2.4 <Lbl>

<Lbl> encloses content that labels other content. Although described as a BLSE in PDF 1.7, this structure element is always an ILSE in practice (this error in the 1.7 specification is corrected in PDF 2.0).

The concept of “label” differs in tagged PDF compared to HTML:

- In tagged PDF, <Lbl> elements are not limited to lists, and are always explicitly contained. <Lbl> structure elements may be used whenever content serves as a label for some other content, for example, “Fig: 123” in a caption for a <Figure>.
- In HTML, labels for list items are implied by the list structure elements, and (possibly) by CSS. The <label> structure element in HTML only applies to labels for an item in a user-interface (e.g., form-fields or buttons).

4.2.4.1 Examples

Example A: (Bullet list)

```
<L> {  
    <LI> {  
        <Lbl> [bullet]  
        <LBody> {content}  
    }  
}
```

Example B: (numbered list)

```
<L> {  
    <LI> {  
        <Lbl> [list item number]  
        <LBody> {content}  
    }  
}
```

Example C: (Table of Contents)

```
<TOC> {  
    <TOCI> {  
        <Lbl> [chapter number]  
        <P> {content}  
    }  
}
```

Example D: (Labels in notes)

See the <Note> element.

The above are examples; other variations are also possible.

4.2.4.2 Creation

No specific guidance provided.

4.2.4.3 Consumption

No specific guidance provided.

4.2.5 <L> (List), (List Item), <LBody> (List Body)

The <L> (list) structure element encloses a sequence of one or more content items with structure elements enclosing each content item. <Lbl> encloses the list item marker for each content item, while <LBody> encloses the content item's content.

For PDF 1.7 (and thus, PDF/UA-1) requires that if present, a list's caption is enclosed in a <Caption> structure element and precedes the structure elements within the captioned <L>.

4.2.5.1 Examples

Example A: (a simple list)

```
<L> {  
    <Caption>  
    <LI>  
    <LI> {  
        <Lbl>  
        <LBody>  
    }  
}
```

Although a canonical form can be derived from ISO 32000-1, the specification does not prohibit other parent-child relationships between the various standard structure elements in a list. As a result, it is recommended that processors be able to handle various forms of lists, including cases in which a <LBody> encapsulates another list that is structurally unrelated to the <LBody>'s parent <L>. Examples of acceptable list structures follow.

Example B: (a canonical multilevel list)

```
<L> {  
    <LI> {  
        <Lbl>  
        <LBody>  
    }  
    <LI> {  
        <Lbl>  
        <LBody>  
    }  
    <L> {  
        <LI> {  
            <Lbl>  
            <LBody>  
        }  
        <LI> {  
            <Lbl>  
            <LBody>  
        }  
    }  
}
```

```

    }
    <LI> {
        <Lbl>
        <LBody>
    }
}

```

Example C: (another commonly encountered, though incorrect, form of nested list)

This model (multilevel list) is borrowed from HTML.

Multilevel list in PDF	Corresponding multilevel list in HTML
<pre> <L> { { <L> [...] } } </pre>	<pre> or in HTML in HTML or in HTML </pre>

Example D: (a list containing another, semantically unrelated 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>  
    }  
}
```

4.2.5.2 *Creation*

For each content item that has a list item marker – such as a bullet or list-numbering – semantic appropriateness requires that the list item marker is enclosed in a <Lbl> structure element.

Semantic appropriateness requires that, apart from the list-item's label itself, the semantic content of each list item be enclosed in an <LBody> structure element, including inline content or arbitrary complex structures.

4.2.5.3 *Consumption*

It is recommended that processors expect to encounter real content as the direct child of an structure element. In such cases, it is recommended that processors treat such content as if it were enclosed in an <LBody> structure element.

4.2.6 <Table>, <TR>, <TH>, <TD>, <THead>, <TBody>, <TFoot>

Table structure types enclose content whose semantics are defined by representation in a matrix of rows and columns.

4.2.6.1 Examples

Example A: (simple data table, without Headers and IDs)

```
<Table> {  
    <TR> {  
        <TD>  
        <TD>  
    }  
    <TR> {  
        <TD>  
        <TD>  
    }  
}
```

Example B: (more complex table, with headers, additionally structured)

```
<Table> {  
    <THead> {  
        <TR> {  
            <TH ID="R1C1_Table" Scope="Column">  
            <TH ID="R1C2_Table" Scope="Column">  
        }  
    }  
    <TBody> {  
        <TR> {  
            <TD>  
            <TD>  
        }  
        <TR> {  
            <TD ColSpan="2">  
        }  
    }  
    <TFoot> {  
        <TR> {  
            <TD>  
            <TD>  
        }  
    }  
}
```

4.2.6.2 *Creation*

Semantic appropriateness requires that:

- irrespective of the use of table authoring tools, table structure elements cannot be used to represent content that does not depend on a matrix of rows and columns for its meaning (e.g., it would be incorrect to use table structure elements in a case where an author has used a table authoring tool simply to align logos with text).
- cells that span rows or columns include appropriate `colspan` and `rowspan` attributes.
- tables spanning multiple pages are structured as a single table. `<TH>` cells in repeated header rows or columns (e.g., in the case of tables that span multiple pages) are marked as artifacts.
- empty cells are always `<TD>` cells, never `<TH>` cells.
- table cells are always part of semantic table structures. An example of a semantically impermissible use of table cells would be an empty row or column separating content that is semantically two (or more) tables.

Rows and columns may use a mix of `<TH>` and `<TD>` cells to allow representation of complex tables.

The primary function of the optional `<THead>`, `<TBody>` and `<TFoot>` structure element types is to aid consuming software in repurposing paginated tables.

4.2.6.3 *Consumption*

AT provides users with information about the relationship between `<TH>` and `<TD>` cells.

4.2.7

The primary purpose of is to demarcate content for the purpose of applying semantic and other attributes. Although the structure element type itself has no inherent semantics, semantic information is provided by attributes employed on a given instance of a structure element type.

 can serve in many capacities but is most useful for subdividing smaller regions of content, such as words within a <P>. may be used as a vehicle for language (**Lang**), replacement content (**ActualText**), alternate description (**Alt**) and expansion (**E**) attributes, and various presentational, layout and other attributes documented in ISO 32000-1, 14.8.5.4 “Layout Attributes”.

4.2.7.1 Examples

Example A:

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

4.2.7.2 Creation

No specific guidance provided.

4.2.7.3 Consumption

No specific guidance provided.

4.2.8 <Note>

<Note> structure elements encompass footnotes and endnotes, and typically operate in conjunction with <Reference> structure elements.

4.2.8.1 Examples

Example A: (Footnote tagged inline):

A possible method for handling <Note> structure elements for which references exist is to locate the note directly following the reference in the logical reading order, as in the following (<Lbl> is most suitable to tag the reference symbol):

```
<P> {  
    <Reference> {  
        <Lbl>  
    }  
    <Note> {  
        <Lbl>  
        <P>  
    }  
}
```

The following example includes a representation of a page-rendering followed by a representation of the structure appearing on the page. In this case the <Note> structure elements are located directly following their respective references in the text.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt.

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.

-
- 1) The first idea is to understand footnotes
 - 2) The second idea is to make them work in PDF

The tagged content of the example:

```
<P> { The first reference {  
    <Reference> {  
        <Lbl> {1} }  
    }  
    <Note> {  
        <Lbl> {1} }  
        <P> {The first idea is to understand footnotes}  
    }  
}
```

```

    for the first footnote on this page, the second reference
    <Reference> {
        <Lbl> {2} }
    }
    <Note> {
        <Lbl> {2} }
        <P> {for the second idea is to make them work in PDF}
    }
    for the second footnote on this page, and so on.
}

```

Example B: (Commonly-encountered work-around)

WARNING: This approach is not official, but well-known and otherwise high-quality agents are known to use this work-around, so the ability to process this circumstance is recommended.

Simply put, it is recommended that developers be prepared to process instances in which a <Note> structure element occurs immediately after the structure element containing the <Reference> structure element that targets the <Note>.

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 A, above, a representation of the structure follows.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod.

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

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

The tagged content of the example:

```

<P> { The first reference {
    <Reference> {
        <Lbl> {1}
    }
    for the first footnote on this page, the second reference
    <Reference> {
        <Lbl> {2}
    }
}

```

for the second footnote on this page, and so on.

```
<Note> {  
    <Lbl> {1}  
    <P> {The first idea is to understand footnotes}  
}  
<Note> {  
    <Lbl> {2}  
    <P> {for the second idea is to make them work in PDF}  
}  
}
```

Example C: (Footnote with link)

```
<P> [content] {  
    <Link> [Destination] {  
        <Link-OBJR>  
        <Lbl> {1}  
    }  
    [more possible content]  
    <Note> [the Destination of the Link] {  
        <Lbl>{1}  
        <P> [content of the footnote] {  
            <Link> [Link back to the reference point]  
            <Link-OBJR>  
        }  
    }  
}  
}
```

NOTE Links for footnotes, in both directions, are not required by PDF/UA, but are recommended.

Example D: (Footnotes / endnotes referenced from multiple locations)

Although ISO 32000-1 does not describe an explicit mechanism to address endnotes referenced from multiple locations, the use case is readily resolved by taking advantage of the fact that the contents of a given footnote's <Lbl> structure element are typically also present inside the corresponding <Reference> structure element.

4.2.8.2 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. It is semantically inappropriate to apply <Note> structure elements to such “notes”. Instead, such cases are best distinguished by a grouping element, as in the following example:

```
<Part> [A grouping element] {  
    <P> [content]  
    <P> [A note pertaining to the content]
```

}

Especially in the context of <L>, <Figure>, <Formula> or <Table> structure elements, <Caption> is usually the semantically appropriate structure element rather than a <Note>.

4.2.8.3 Creation

It is strongly recommended that <Note> structure elements only be used for explicitly referenced content such as footnotes, endnotes, or table or figure notes (a type of footnote), irrespective of pagination.

As shown in Example B, in PDF 1.7, association between a <Reference> and its <Note> structure element may most readily be accomplished by ensuring that the <Lbl> in a <Reference> structure element matches the <Lbl> in the corresponding <Note> structure element. However, this approach is only fully reliable when:

- the label is not repeated elsewhere in the document, or
- the same-labelled <Note> follows the <Reference> in the logical reading order.

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

4.2.8.4 Consumption

Following Example B, to find notes for a given reference, from a <Reference> structure element, search forward in the logical reading order for a <Note> structure element containing a <Lbl> structure element with the same content as the <Reference> structure element's <Lbl>.

Optionally, provide a mechanism to hide <Note> content unless visited from a <Reference>. In general, AT conforming to PDF/UA is obliged to provide functionality that presents <Reference> and <Note> structure elements in a suitable way, for example:

- informing the user when a <Reference> is encountered
- providing navigation to the associated <Note> content
- allowing a user to return to the <Reference> after visiting the matching <Note>
- skipping <Note> structure elements when reading text sequentially
- handling <Note> structure elements as ILSE, BLSE or grouping elements, depending on usage.

NOTE 1 <Reference> and <Note> structure elements do not exist in HTML.

NOTE 2 Due to limitations in the underlying PDF specification, although PDF/UA-1 requires ID attributes on <Note> structure elements, they provide no added value to consuming processors.

A link on a reference may target a Destination (ISO 32000-1, 12.3.2 “Named Destinations”), however, use of this feature by AT requires detailed session navigation. Absent such navigation, it is recommended to allow users to return to the <Reference> structure element in the text (this applies to all document-internal links; not just in the Reference/Note context).

4.2.9 <Reference>

<Reference> structure elements encompass content that refers to other content, typically, the contents of respective <Note> structure elements (see 4.2.8, “<Note>” for additional guidance in that context) or within an <Index> or <TOC>. It is mostly often used as an inline element.

<Reference> elements do not, by themselves, imply any specific interactivity. If interactivity is desired, Link annotations (and their respective <Link> structure elements) are needed. See "Link within Reference" in this subclause.

4.2.9.1 Examples

Example A: (usage within a paragraph)

```
<P> [content] {  
    <Reference> [content of reference, e.g. "1"]  
}  
possible more content  
}
```

Example B: (usage within a <TOC>)

```
<TOC> {  
    <TOCI> {  
        <Reference> {The content of a headline}  
    }  
}
```

Example C: (usage within a <TOC> including <Reference> & <Link>)

(This example is copied from clause 4.1.4)

```
<TOC> {  
    <TOCI> {  
        <P> {  
            <Reference> {  
                <Link> {  
                    <Link-OBJR>  
                    <Lbl> [In cases where the TOCI is numbered] {  
                        1.  
                    }  
                    <Span> {  
                        Introduction ..... page 5  
                    }  
                }  
            }  
        }  
    }  
}
```

}

4.2.9.2 Creation

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

- A <Reference> structure element containing a <Lbl> structure element is assumed to point to a footnote / endnote, bibliography reference or other such target. In such cases, the contents of the <Lbl> structure elements in both the originating and target structure elements exactly match, and in addition, there is no footnote / endnote present between a <Reference> structure element and its corresponding footnote / endnote with the same <Lbl>.
- A <Reference> structure element not containing a <Lbl> structure element is assumed to be a cross-reference. This also applies to <Reference> structure elements contained in <TOCI> or <Index> structure elements as defined in ISO 32000-1 Table 333.

4.2.9.3 Consumption

Based on the creation practice proposed above it is always possible to determine the <Note> corresponding to a given footnote or endnote's <Reference> structure element, as follows:

- When encountering a footnote or endnote <Reference> structure element, use the contents of the <Lbl> structure element 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 the originating <Reference> from a given <Note>'s <Lbl> by means of scanning backwards in the document to find the matching <Lbl> in a <Reference> structure element.

NOTE: <Reference> and <Note> structure elements do not exist in HTML.

An AT wishing to take advantage of <Reference> can offer a means of round-trip navigation from a <Reference> structure element to the matching <Note> structure element. It is recommended that such implementations take into account that more than one <Reference> may point to the same <Note> structure element.

<Link> within <Reference>

<Reference> elements can include <Link> structure elements. The nesting of <Link> elements within <Reference> elements is expressed by some APIs with two announcements of a link instead of one. This behavior is semantically inappropriate.

4.2.10 <BibEntry>

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

Support for this structure element by AT is not anticipated.

4.2.10.1 *Examples*

None provided.

4.2.10.2 *Creation*

No specific guidance provided.

4.2.10.3 *Consumption*

No specific guidance provided.

4.2.11 <Code>

Intended to semantically identify code examples, this structure element serves to indicate that enclosed content would preferably be represented precisely; without further modification (for example: justification, cleanup of white space). However, use of this structure element does not imply that extraction would result in usable code.

Support for this structure element by AT is not anticipated.

4.2.11.1 *Examples*

None provided.

4.2.11.2 *Creation*

No specific guidance provided.

4.2.11.3 *Consuming*

No specific guidance provided.

4.2.12 <Link>

The <Link> structure element typically associates content and actionable link annotation(s).

4.2.12.1 Examples

Example A:

```
<P> { Visit the
    <Link> {
        [link text, eg. URL of the website]
        OBJR-xxx (https://...)
    }
today!
}
```

4.2.12.2 Creation

PDF/UA-1 does not require that <Link> structure elements enclose content.

Multiple link annotations enclosed in a single <Link> structure element

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

Support for QuadPoints allows for more reliable experiences with interactive viewers. Without QuadPoints, very common cases (e.g. URLs breaking across one or more lines) may lead to multiple annotations and undesirable user experiences.

Although in general OBJRs may appear at any location within the <Link> element, for links that span pages, it is recommended that all OBJRs be next to each other in the logical order, as in the following example:

```
<P> {
    <Link> {
        [PAGE 7]
        https://www
        OBJR 1
        [page break]
        [PAGE 8, link continues]
        OBJR 2
        .pdfa.org
    }
}
```

The Contents key

Although required by PDF/UA-1, in practice, many common tools do not provide easy access to the **Contents** key of link annotations. Many (or most) current-generation AT and other software do not process this key. Typical workarounds to achieve formal PDF/UA conformance include automated population of the **Contents** key from (for example) the **Alt** key in the <Link> structure element.

*NOTE: Relaxation of the **Contents** key requirement in PDF/UA-1 is anticipated in PDF/UA-2.*

4.2.12.3 Consumption

As discussed above, it is recommended to be prepared for cases in which multiple links are encoded representing a single (logically speaking) link object (e.g., a single semantic link spanning two lines of text).

In the case where a single <Link> structure element encloses multiple link annotations, and where all link annotations have identical targets, and if it is semantically correct to do so, representing a single link to the user is likely to deliver a better user experience.

4.2.13 <Annot>

The <Annot> structure element encloses annotations other than links and widgets (see 4.3.3, “<Form>”).

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.

4.2.13.1 Examples

Example A:

```
<Annot> {  
    Content  
    OBJR [pointing to <Annot> of type "Highlight"]  
}
```

Annotations may contain video, 3D and other non-PDF content, the accessibility of which is outside the scope of this Guide. It is recommended that developers seek out appropriate guidance on ensuring these formats are accessible.

4.2.13.2 Creation

A challenge can arise when content to be marked up by a markup annotation is not already represented by a structure element, and thus cannot be directly associated with the annotation. In such a case it is often semantically appropriate to enclose the marked-up content 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.

4.2.13.3 Consumption

It's vital to avoid a case where consumers experience two renditions of the same alternative content. Accordingly, it is recommended that processors be sensitive to the use of the **Contents** key in the annotation as well as the use of the **Alt** property on the enclosing structure element.

When presenting marked-up content to a user, it is recommended that the following aspects 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, it is recommended that processors also make available annotation properties, for example: author, subject, status, date/time, checkmark.

4.2.14 <Quote>

<Quote> encloses quoted content within a paragraph (inline element). See also <BlockQuote> a block-level element that encloses longer portions of quoted content.

4.2.14.1 Examples

Example A:

A quote within a paragraph:

```
<P> { some text {  
    <Quote> {quoted text}  
  }  
  more text  
}
```

4.2.14.2 Creation

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

Where quoted content does not exist inside a paragraph or other block-level structure element, the <BlockQuote> structure element is used.

4.2.14.3 Consumption

It is recommended to present quoted content such that a consumer can distinguish between quoted and unquoted content.

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

4.2.15 <Ruby>, <RB>, <RT>, <RP>, <Warichu>, <WT>, <WP>

These structure elements represent specific cases in Japanese typography.

No guidance provided at this time.

4.3 Illustration elements

4.3.1 <Figure>

A <Figure> element encloses content that, however encoded, represents non-textual visual content (e.g., a photograph, an illustration, etc.).

PDF 1.7 does not specify a mechanism to associate <Figure> structure elements with their <Caption> structure elements, or associate multiple figures together, or apply a caption to multiple figures.

In the context of <Figure> structure elements, it is recommended to locate the <Caption> structure element following the <Figure> structure element, as this practice ensures a reasonable context for the <Caption> is provided to users of relatively basic consumption software.

NOTE In principle, other grouping elements such as <Part>, <Sect>, etc. could be used. In most cases, however, these would be semantically inappropriate for the purpose of grouping <Figure> structure elements.

4.3.1.1 Examples

Example A: (A <Figure> with a <Caption> following the <Figure>)

```
<Figure>      [A structure element enclosing an actual image]
               {CONTENT}    [The actual image or illustration]
<Caption>     [The Figure's caption]
```

Example B: (A <Figure> with a <Caption> preceding the <Figure>)

```
<Caption>     [The Figure's caption]
<Figure>      [A structure element enclosing an actual image]
               {CONTENT}    [The actual image or illustration]
```

Example C: (Multiple <Figure> elements with <Caption> elements)

```
<Figure>      [A structure element enclosing an actual image]
{CONTENT}     [The actual image or illustration]
<Caption>     [The first figure's caption]
<Figure>      [A structure element enclosing an actual image]
               {CONTENT}    [The actual image or illustration]
<Caption>     [The second figure's caption]
```

Example D: (<Figure> without <Caption>)

While it's preferred for <Figure> structure elements to include <Caption> elements there are many use cases where a <Figure> does not have a <Caption> (e.g.: a logo, title-page image, in-line graphics such as smileys, symbols or other illustrative content). Such cases may be handled as in the example:

```
<Figure> {CONTENT}    [The actual image or illustration]
```

4.3.1.2 Creation

No specific guidance provided.

4.3.1.3 *Consumption*

In the case of a <Figure>'s <Caption> element, it is recommended that both the <Caption> and the **Alt** property for the <Figure> be presented to the user.

4.3.2 <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; it 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.

4.3.2.1 Examples

Example A: (Formula as Text)

```
<Formula> {2 + 2 = 4}
```

Example B: (Formula as a Figure)

```
<Formula> {  
  <Span ActualText="2 + 2 = 4">  
}
```

4.3.2.2 Creation

For formulas represented with raster and/or vector images, it is semantically appropriate to use <Formula> rather than a <Figure> structure element since these objects, whatever their encoding, are in a semantic sense, formulas, not figures.

Consider using MathML as (in PDF 1.7) custom structure types mapped to . For table-like structures, consider leveraging **Table** structure elements. Map the <Math> structure element to <Formula>.

Note that PDF/UA requires that a <Formula> structure element include an **Alt** attribute (PDF/UA-1, 7.7).

It is recommended that although content inside a <Formula> structure element may make use of an **ActualText** property, it's preferred to do so only on very small pieces of content, such as ligatures. It is recommended that the **ActualText** property be on the respective structure element, not the <Formula> structure element.

4.3.2.3 Consumption

Due to ambiguities in English, where “figure” may be understood to refer to a mathematical formula, quite often the <Figure> structure element is used to enclose math, which is semantically incorrect. It is recommended that processors be prepared to encounter this case.

4.3.3 <Form>

<Form> structure elements are used to enclose form field widgets (ISO 32000-1, 12.5.6.19), identifying and ordering them for consumption by AT. Some types of PDF form fields have a single interactive object (in PDF parlance, a “widget annotation”) associated with them (e.g., pushbutton or text form field), others, as in the case of radio button groups, can have several interactive objects associated with them.

4.3.3.1 *Form fields vs. widget annotations*

Each <Form> element encloses a single widget annotation. 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 together widget annotations belonging to the same form field; instead, the data model relies on the **T** key (form field name).

Regarding non-interactive forms, see 5.3, “PrintField attributes”.

4.3.3.2 *Labeling form fields*

ISO 32000 does not provide any specific mechanism for identifying form field labels in page content. Such association can only be provided by way of context and logical ordering (i.e., placement of a <Lbl> structure element next to the form field and/or the <Form> elements associated with it. Accessible PDF forms are those in which both logical and Acroform structures are in alignment.

4.3.3.3 *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.

The Opt entry

The optional **Opt** key 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”.

A simple radio-button example:

Prefix:
☒ Ms. ☐ Mr.

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

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>

4.3.3.4 Creation

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

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

4.3.3.5 *Consumption*

It is recommended that interactive viewers encountering forms using **PrintField** attributes indicate that such forms are “read only”.

It is recommended that readers avoid an approach in which the user has only form fields presented, and by implication, may miss other content.

How to find out the meaning of each radio button

It is recommended that each widget in a radio button form field contain two entries in the appearance dictionary’s **N** (normal) key in which one of these entries has the name “Off”; the other key 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** key in the radio button form field. In the case of missing **V** key the value is “Off”.

5 Attributes and properties

5.1 Layout attributes

Layout (position, color, etc.) is not encoded into PDF content streams in a way that strongly associates stylistic properties with content. It is semantically inappropriate to fail to include layout attributes necessary to convey the semantics of a given use of colour, contrast, format or layout.

5.1.1 Standard layout attributes

Exhaustive conformance requirements for the use of standard layout attributes (see ISO 32000-2, Tables 378, 379 and 380) are described in Table 1, “Requirements for standard layout attributes” using the following terms:

- “Required” The attribute always has a semantic purpose. Semantic appropriateness requires that it be present in all cases of applicable structure elements when the relevant semantic property is present in the content and differs from the default value (if any)
- “Required if semantic” The attribute often has a semantic purpose. Semantic appropriateness requires that it be present in all cases of applicable structure elements when the relevant semantic layout property is present in the content; is intended to have semantic significance; and differs from the default value (if any)
- “Not required” The attribute is optional, rarely has a semantic purpose and is never required to be present on any structure element

Table 1: Requirements for standard layout attributes

Structure elements	Attribute	Conforming usage implied by PDF/UA-1
Any structure element	Placement	Not required Default value: <i>Inline</i>
	Writing Mode	Required EXAMPLE: Text written from right to left. Default value: <i>LrTb</i>
	BackgroundColor	Required if semantic EXAMPLE: Background color of a TD structure element identifying a group of cells. Default value: (none)
	BorderColor	Required if semantic EXAMPLE: The border of a box in which the box’s appearance denotes the significance of the content (such as an alert).

Structure elements	Attribute	Conforming usage implied by PDF/UA-1
		Default value: (Current text fill color in effect)
	BorderStyle	Required if semantic EXAMPLE: Lines used to represent relationships in an organization chart. Default value: <i>None</i>
	BorderThickness	Required if semantic EXAMPLE: A table cell where thicker lines are used to denote that the cell contains a summation of other cells' values. Default value: (none)
	Color	Required if semantic EXAMPLE: Use of colour in a chart legend. Default value: (Current text fill color in effect) NOTE Despite the default value for Color, to facilitate access to this attribute, it is useful to set this value in all semantic cases.
	Padding	Not required
Any BLSE; ILSEs with Placement other than Inline	SpaceBefore	Required if semantic EXAMPLE: Visual grouping Default value: 0
	SpaceAfter	Required if semantic EXAMPLE: Visual grouping Default value: 0
	StartIndent	Required if semantic EXAMPLE: Different levels of indentation of lines for code in a programming language. Default value: 0

Structure elements	Attribute	Conforming usage implied by PDF/UA-1
	EndIndent	Required if semantic EXAMPLE: Lines of code. Default value: 0
BLSEs containing text	TextIndent	Not required
	TextAlign	Not required
Figure, Formula and Table	BBox	Required EXAMPLE: A collection of paths whose semantic value arises from their consideration as a whole. NOTE: Current-generation AT often relies on this attribute.
	Width	Not required
	Height	Not required
TH (Table header), TD (Table data)	Width	Not required
	Height	Not required
	BlockAlign	Not required
	InlineAlign	Not required
	TBorderStyle	Required if semantic EXAMPLE: The border of a cell in which the cell's appearance denotes the significance of its content (such as distinguishing cells in certain AT applications). Default value: <i>None</i>
	TPadding	Not required
Any ILSE; BLSEs containing ILSEs or containing direct or nested content items	LineHeight	Not required
	BaselineShift	Required if semantic EXAMPLE: Super or subscripts. Default value: 0
	TextDecorationColor	Required EXAMPLE: Red underline used to indicate authorship of an edit.

Structure elements	Attribute	Conforming usage implied by PDF/UA-1
		<p>Default value: (Current fill color in effect)</p> <p>NOTE Despite the default value for TextDecorationColor, to facilitate access to this attribute, it is useful to set this value in all semantic cases.</p>
	TextDecorationThickness	<p>Required if semantic</p> <p>EXAMPLE: Thick underline used to indicate misspelling.</p> <p>Default value: (Current stroke thickness in effect)</p> <p>NOTE Despite the default value for TextDecorationThickness, to facilitate access to this attribute, it is useful to set this value in all semantic cases.</p>
	TextDecorationType	<p>Required</p> <p>EXAMPLE: Strikethrough.</p> <p>Default value: <i>None</i></p>
	RubyAlign	Not required
	RubyPosition	Not required
	GlyphOrientationVertical	<p>Required if semantic</p> <p>EXAMPLE: Representation of portrait or landscape orientation using the letter "A".</p> <p>Default value: <i>Auto</i></p>

5.1.2 Standard layout attributes specific to inline-level structure elements

Requirements for the use of standard layout attributes (see ISO 32000-2, Tables 381, 382, 383) as described in Table 2, “Requirements for standard layout attributes specific to inline-level structure elements”.

Table 2: Requirements for standard layout attributes specific to inline-level structure elements

Structure elements	Attribute	Conformance requirements
Grouping elements containing columns	ColumnCount	Not required
	ColumnGap	Not required
	ColumnWidths	Not required
L	ListNumbering	Required if the value is not <i>None</i> EXAMPLE: Ordered lists. Default value: <i>None</i>
PrintField	Role	Required EXAMPLE: A pre-filled (possibly flattened) form field. Default value: None specified
	Checked, checked	Required EXAMPLE: A pre-filled (possibly flattened) form field. Default value: <i>off</i>
	Desc	Required EXAMPLE: A pre-filled (possibly flattened) form field. Default value: None specified

5.2 List attributes

5.2.1 ListNumbering

ISO 32000-1 does not address the concept of continued lists (multiple list fragments with other block level elements between fragments); semantic encoding of this construct is unavailable prior to 32000-2.

Arbitrary labels use the value *None*. Accordingly, it is recommended that processors intending conversion to HTML consider that a **ListNumbering** attribute value of *None* strongly implies use of the contents of the <Lb1> structure elements in the list to represent the list's labels, akin to HTML's description lists (<dl>). An example:

```
<Lb1> {Day 1:}
<Lb1> {Day 2:}
```

5.3 PrintField attributes

PrintField attributes define PDF's accessibility mechanism for non-interactive PDF forms (see ISO 32000-1, 12.7.9, "Non-interactive forms"). Such forms may have originally contained interactive fields such as text fields and radio buttons but were then converted into non-interactive PDF files, or they may have been designed to be printed out and filled in manually.

Radio buttons and list boxes are not fully accommodated in ISO 32000-1's definition of **PrintField** attributes. These attributes are entirely overhauled in ISO 32000-2.

5.4 Table attributes

5.4.1 Scope, Headers and IDs

For ordinary tables the **Scope** attribute is generally the easiest approach to establishing the table's structure. In keyboard navigation, where **Scope** is sufficient, right arrow = Row, down arrow = Column

Column 1 header	Column 2 header	Column 3 header
Row 2 header	Column 2, Row 2 data	Column 3, Row 2 data
Row 3 header	Column 2, Row 3 data	Column 3, Row 3 data
Row 4 header	Column 2, Row 4 data	Column 3, Row 4 data

For complex table structures, including subdivisions within tables, **Headers** and **ID** attributes are generally necessary in order to correctly represent the table's structure by way of the **Headers** entry in each **TH** cell enumerating the **IDs** of the table cells (**TDs** and **THs**) it heads.

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

	Column 2 header	Column 3 header
Row 2 header	Column 2, Row 2 data	Column 3, Row 2 data
Row 3 header	Column 2, Row 3 data	Column 3, Row 3 data
Row 4 header	Column 2, Row 4 data	Column 3, Row 4 data
	Column 2 subheader	Column 3 subheader
Row 6 header	Column 2, Row 6 data	Column 3, Row 6 data
Row 7 header	Column 2, Row 7 data	Column 3, Row 7 data
Row 8 header	Column 2, Row 8 data	Column 3, Row 8 data

5.4.2 Summary attribute

It is recommended that use of this attribute 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 is recommended that such 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 it is recommended that the practice be limited to such cases.

5.5 Commonly-used properties of content

PDF's content properties may be combined to accommodate a variety of use-cases.

The four keys defining properties of content can appear in both the property list dictionary (for a marked content sequence with a **Span** tag) and the structure element dictionary (for structure elements).

Property	Purpose
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.
Alt	(alternate description) Provides descriptive information for content with a substantial non-textual aspect.
ActualText	(replacement text) The text representation of text that is not encoded as text.
E	(expansion of abbreviations and acronyms) Provides expanded text.

These properties are discussed in the following clauses.

5.5.1 Lang

The **Lang** property can be used to provide the assigned language of content. A language identifier is a language code whose specificity (e.g., *fr* vs. *fr-ca*) may vary.

Lang exists at four levels in a PDF document:

- Document level: a **Lang** entry in the document's catalog dictionary
- Logical structure level: a **Lang** property 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 document's catalog dictionary establishes the default language for the entire document, including page content, metadata and text strings within annotations.

A **Lang** property 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** 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.

5.5.1.1 *Example*

ISO 32000-1 provides adequate examples.

5.5.1.2 *Creation*

Although the **Contents** key in annotations is unevenly supported, it is the correct means of providing an alternate description for annotations.

Placing the same description in both the **Contents** key and the **Alt** property is not recommended, as this can lead some implementations to present the same alternative description twice.

5.5.1.3 *Consuming*

Lang governs all attributes of the structure element on which it appears, and all enclosed content. When **Lang** occurs at a document level (i.e., as an entry in the catalog dictionary), it serves as the ‘base language’ for the entire document, including outlines (bookmarks) and metadata, except where overridden by a content-specific (structure element or marked content sequence) use of **Lang**.

5.5.2 *Alt (alternate description)*

Alt 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.

Text can be encoded as an image object. In such cases, the semantically appropriate property for accessibility purposes is **ActualText** providing the text represented by the image; the **Alt** property is semantically inappropriate in this case and should be not defined (See 5.5.3, “ActualText”).

5.5.2.1 *Typical usage*

While most commonly used on the <Figure> structure element, **Alt** is used with any content that is mostly not text-based, including cases where that content consists of several distinct graphics objects. Examples include:

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

5.5.2.2 *Nested structures*

Illustrations can include subdivisions that can be considered as individual illustrations in their own right, and for which. In such cases an **Alt** property is semantically appropriate for both the main illustration and its components. Examples include:

- Diagrams of sub-assemblies
- Building plans
- Maps including countries and states

5.5.2.3 Creation

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.

5.5.2.4 Consuming

In cases where **Alt** and **ActualText** properties both occur on the same structure element, both must be available to the user.

5.5.3 ActualText

For this clause, the term “text content” refers to content that is visually perceived as text regardless of the content’s actual encoding as text, image or vector objects, clipping paths, masks, or any combination thereof.

ActualText is used for content that is normally being perceived as text but is not encoded as text. Typical examples include a small image representing a single word, or a vector object representing a single character. **ActualText** makes it possible to associate the text that would normally be perceived by a sighted user with respective objects. By implication, this makes it possible to derive a text-only representation of the content, e.g., for use by a screen reader, search engine or other text-consuming technology.


This property is useful in a variety of circumstances, including:

- An image 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)
 - When deciding between applying **ActualText** in marked content vs. a structure element

While it is technically acceptable to use **ActualText** as a property of any structure element, it is recommended to only use the **ActualText** property on structure elements of type , or for Span marked content sequences.

Example A:

ActualText property is associated with a structure element:

```
<P> {  
    <Span ActualText='T'> {  }  
    he PDF format  
}
```

ActualText property is associated with a Span marked content sequence:

```
/P <</MCID 0 >> BDC
  /Span <</ActualText (T) >> BDC
  /Im1 Do
  EMC
  ...
  (he PDF format) Tj
EMC
```

In all cases where a structure element is used, a Span marked content sequence is an acceptable substitute.

Example B:

```
<TR> {
  <TD> {
    <Span ActualText="1">
      {image of a numeral '1'}
    }
  <TD> {2}
}
```

NOTE Widely used reading programs and ATs tend to present the first table cell in the alternative approach as just text, ignoring the fact that it is a table cell, and thus presenting a broken table structure.

Example C:

```
<P> { The German word for "printer" is:
  "Dru {
    <Span ActualText='ck'>{k-k}
  }
  er"
}
```

NOTE This example assumes a line-break in the middle of the word "Drucker", which implies that the "ck" must be written as "k-k".

5.5.3.1 Creation

ActualText can be used on marked content sequences or on structure elements. When used on structure elements, it is recommended that the attribute not be used on structure element types such as <Figure>, <TH> or <TD>. When the need arises to associate content in a <Figure>, <TH> or <TD> with **ActualText**, it is recommended that a structure element be added inside the <Figure>, <TH> or <TD> structure element, and the **ActualText** property be associated with that structure element.

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

In cases where it isn't possible to provide a ToUnicode table, the **ActualText** property 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, it is recommended that software provide text content as text objects in the page description (including white-space characters). If this is not possible or excessively difficult, use (in the following order):

1. **ActualText** on the content, or
2. **ActualText** on an empty structure element. It is semantically appropriate to only use this approach 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 rather than on the marked content sequence.

NOTE 1 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 can be superimposed as text objects in render mode 3 (invisible text).

*NOTE 2 Use of **ActualText** is limited to text that would otherwise be contained within a single structure element.*

Scanned pages

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

As with any PDF/UA-1 conforming file, it is semantically appropriate to contain each semantically-significant figure on the scanned page within <Figure> structure elements. This will necessitate some means of selective inclusion of that portion of the page in the page description.

5.5.3.2 Consumption

In the (inadvisable) case of an **ActualText** property on a <TD> or <TH> structure element, some popular APIs recognize an **ActualText** property by replacing the <TD> or <TH> structure element, respectively, with the ActualText's property's value. However, semantic appropriateness requires that all semantics be retained, although the means of doing so are implementation-dependent.

5.5.4 E

PDF/UA-1 does not mandate use of the **E** property. However, in some cases, or for certain audiences, the **E** property can be useful (e.g., for providing fully spelled-out text for an abbreviation).

In general, it is recommended that such fully spelled-out text be contained in the document as ordinary content in the vicinity of the abbreviation.

Example:

```
<Span E='Portable Document Format'> {  
  PDF  
}
```


5.5.4.1 Creation

None provided.

5.5.4.2 Consumption

It is recommended that any viewer be able to indicate the presence of an **E** property, and – at the discretion of the user – present its contents.

6 Text characteristics

6.1 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 from the surrounding text, and for other purposes.

Tagged PDF in PDF 1.7 does not include an explicit mechanism to express superscript or subscript semantics. Although the **BaselineShift** attribute may be used as a substitute, it cannot be considered a fully reliable mechanism, as text could be shifted for other reasons.

6.2 Symbolic characters

It is semantically appropriate for consuming software to 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, ZapfDingbats, Wingdings.

Where symbolic characters (such as Wingdings glyphs) are used, use ToUnicode table entries, or **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. The use of the Unicode PUA is discouraged, as no predefined meaning is associated with Unicode values in the PUA.

Where symbolic characters are used as a figure, use the <Figure> structure element with appropriate alternate text.

7 Other features of PDF

7.1 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 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 in the logical structure. Here we offer some general principles for handling such cases.

7.1.1 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 structure element 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 is generally via a separate user interface, not by artificially forcing the digital signature into the logical reading order.

7.1.2 Requirements for field appearances

Since objects within a digital signature appearance stream cannot be structured, 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** property on the <Form> structure element for the digital signature suffices for PDF/UA-1 conformance.

7.2 Page open options

It is recommended that highly structured and/or longer documents include outlines (ISO 32000-1, 12.3.3) and that outlines follow the headings present in the document. For documents including outlines, it is recommended that the default view of the document displays the outline entries.

NOTE Outlines are commonly known as "bookmarks" due to longstanding viewer conventions.

8 Editing tagged PDF files

Consideration is due when merging PDF documents with different language settings, metadata, or other features.

Deleting pages, splitting documents, inserting pages and similar operations require that structure elements (as well as bookmarks or internal link destinations) connected with page content objects be handled appropriately.

When inserting pages into a tagged PDF document, care must be taken to reconcile the structure elements associated with the inserted pages with that of the target document

It is semantically appropriate to have the deletion of content result in the deletion of corresponding structure elements.

Annex A: The PDF/UA flag

An example of a complete PDF/UA flag:

```
<?xpacket begin="" id="W5M0MpCehiHzreSzNTczkc9d"?>
<x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="Adobe XMP Core 5.6-c015
91.163280, 2018/06/22-11:31:03" >
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    <rdf:Description rdf:about=""
      xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:pdfuaid="http://www.aiim.org/pdfua/ns/id/"
    <dc:title>
      <rdf:Alt>
        <rdf:li xml:lang="x-default">PDF/UA Document</rdf:li>
        <rdf:li xml:lang="en">PDF/UA Document</rdf:li>
      </rdf:Alt>
    </dc:title>
    <pdfuaid:part>1</pdfuaid:part>
  </rdf:Description>
</rdf:RDF>
</x:xmpmeta>
<?xpacket end="w"?>
```

Annex B: PDF 2.0

Although intended for implementers of PDF 1.7 and PDF/UA-1, this Annex provides limited guidance for implementers looking forward to PDF 2.0 (ISO 32000-2) and PDF/UA-2. This Annex is not intended as a guide to Tagged PDF syntax in PDF 2.0 and cannot be used as a substitute for ISO 32000-2 (PDF 2.0).

A.1 Differences between PDF 1.7 and PDF 2.0

The basic data model for Tagged PDF remain unchanged between PDF 1.7 and PDF 2.0. However, there are many changes in the use of objects defined in Tagged PDF, including, but not limited to:

- A new namespace mechanism allows for the specification of tag sets, including sets external to ISO 32000
- Introduces a new PDF 2.0 namespace:
 - New standard structure types
 - Certain standard structure types defined in PDF 1.7 are not present in the PDF 2.0 namespace
 - Changed definitions for many standard structure elements
 - Possible hierarchical relationships between standard structure elements are precisely identified
- Defines the PDF 1.7 namespace:
 - Based on the PDF 1.7 tagset defined in ISO 32000-1:2008
 - Allows usage of PDF 1.7 elements in PDF 2.0
- The PDF 1.7 namespace is the default namespace for PDF 2.0 documents
- Introduces MathML as a first-class namespace for PDF 2.0
- Concepts such as artifacts, alternate descriptions and replacement text are improved
- New artifact types and subtypes are added
- Pronunciation hints may be added

A.2 Namespaces and standard structure types

Many PDF documents are authored by conversion from other formats, many of which have rich structures and content with their own structures. the namespace mechanism introduced in PDF 2.0 allows one or more of these externally-defined namespaces to be specified as being used within the document (see ISO 32000-2, 14.7.4.2, "Namespace dictionary").

Examples of such namespaces that might be used in a PDF file include, among others:

- Chemical Markup Language (CML)
- Standard Music Description Language (SMDL)

A.3 Investing in PDF 2.0 while supporting PDF 1.7

A.3.1 General

The PDF 1.7 tagset is fully available in PDF 2.0; rules for using PDF 1.7 tags in a PDF 2.0 context are defined in PDF 2.0.

PDF 2.0 includes complete information on the allowable parent-child relationships between all structure element types. Although these rules apply to PDF 2.0, they also represent best-practice for PDF 1.7 implementations. Accordingly, it is strongly recommended that developers implementing PDF 1.7 also avail themselves of PDF 2.0, and especially Annex L therein.

A.3.2 Some changes in common structure element types

For those implementers considering extending their support for Tagged PDF to PDF 2.0, changes to some standard structure element types may affect choices made when implementing Tagged PDF according to ISO 32000-1. The set of structure element types discussed in this subclause is not intended to be exhaustive.

A.3.2.1 <H1> – <H6>

Since headings commonly appear in Tables of Contents, and since document titles do not normally appear in Tables of Contents, a PDF 2.0-safe approach would be to use <Title> (which is defined in PDF 2.0) mapped to the <P> structure type. Upgrading this document to PDF 2.0, therefore, would simply require deletion of this role map.

NOTE PDF 2.0 adds a <Title> structure type for the purpose of tagging document titles, 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 document's title. If a <Title> structure element encloses the title content as a custom structure element, role-mapped to <P>, then conversion to PDF/UA-2 can be achieved by simply removing the role-mapping.

A.3.2.2 <Caption>

PDF 2.0 updates the description of <Caption> as follows:

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.

A structure element is understood to be "captioned" when a <Caption> structure element exists as an immediate child of that structure element. The <Caption> shall be the first or the last structure element inside its parent structure element. The number of captions cannot exceed 1.

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

A.3.2.3 <Note> and <Reference>

PDF 2.0 makes it possible to explicitly associate references with notes via the new **Ref** key in the structure element dictionary. Additionally, structure destinations on link annotations are also possible in PDF 2.0.

Although the standard structure namespace for PDF 2.0 does not define a <Reference> standard structure element, PDF 2.0 readers supporting Tagged PDF are required to support the PDF 1.7 structure element set as the default namespace.

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 standard structure namespaces defined in PDF 2.0.

Bibliography

PDF/UA-1 Technical Implementation Guide: Understanding ISO 14289-1 (PDF/UA-1)

http://www.aiim.org/Global/AIIM_Widgets/Community_Widgets/Technical-Implementation-Guide

PDF/UA-1 Technical Implementation Guide: Understanding ISO 32000-1 (PDF 1.7)

http://www.aiim.org/Global/AIIM_Widgets/Community_Widgets/Technical-Implementation-Guide-32000-1

Achieving WCAG 2.0 with PDF/UA

http://www.aiim.org/Global/AIIM_Widgets/Community_Widgets/Achieving_WCAG

The Matterhorn Protocol, 2022, PDF Association

<https://www.pdfa.org/community/pdf-ua-technical-working-group/>

PDF/UA Reference Suite, 2022, PDF Association

<https://www.pdfa.org/community/pdf-ua-technical-working-group/>