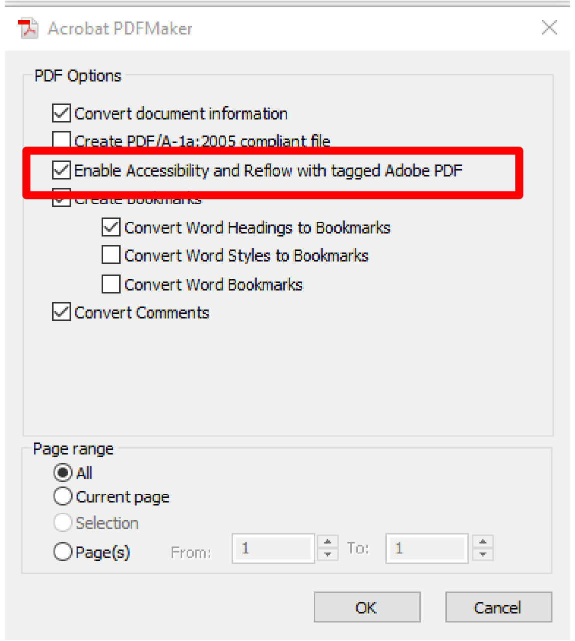
Tags in PDF, and Their Effect on Reconstruction

Understanding the Power of Solid Framework

What on Earth are “Tagged Tables”?

- The PDF specification allows “tags” to be included that specify the type of data that “ink” represents
- The “Save As Adobe PDF” option in Word 2016, for example, allows the user to choose to do this



PDF Options

☑ Convert document information

☐ Create PDF/A-1a:2005 compliant file

☑ Enable Accessibility and Reflow with tagged Adobe PDF

☑ Create Bookmarks

☑ Convert Word Headings to Bookmarks

☐ Convert Word Styles to Bookmarks

☐ Convert Word Bookmarks

☑ Convert Comments

Page range

● All

○ Current page

○ Selection

○ Page(s) From: 1 To: 1

The intention is to allow the data to be reflowed in a sensible way and to add context to the file

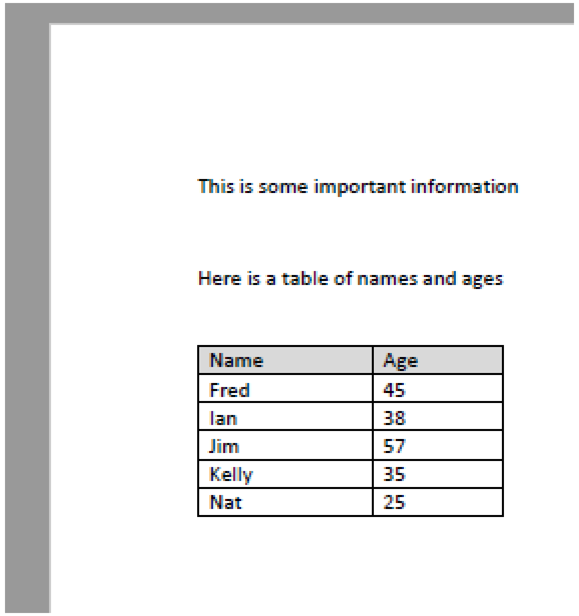
What Adobe says about structure tags

“Document Structure Tags and Proper Reading Order

To read a document’s text and present it in a way that makes sense to the user, a screen reader or other text-to-speech tool requires that the document be structured. Document structure tags in a PDF define the reading order and identify headings, paragraphs, sections, tables and other page elements. The tags structure also allows for documents to be resized and reflowed for viewing at larger sizes and on mobile devices.”

https://www.adobe.com/accessibility/pdf/pdf-accessibility-overview.html

At first sight this has no affect on how the PDF appears when opened

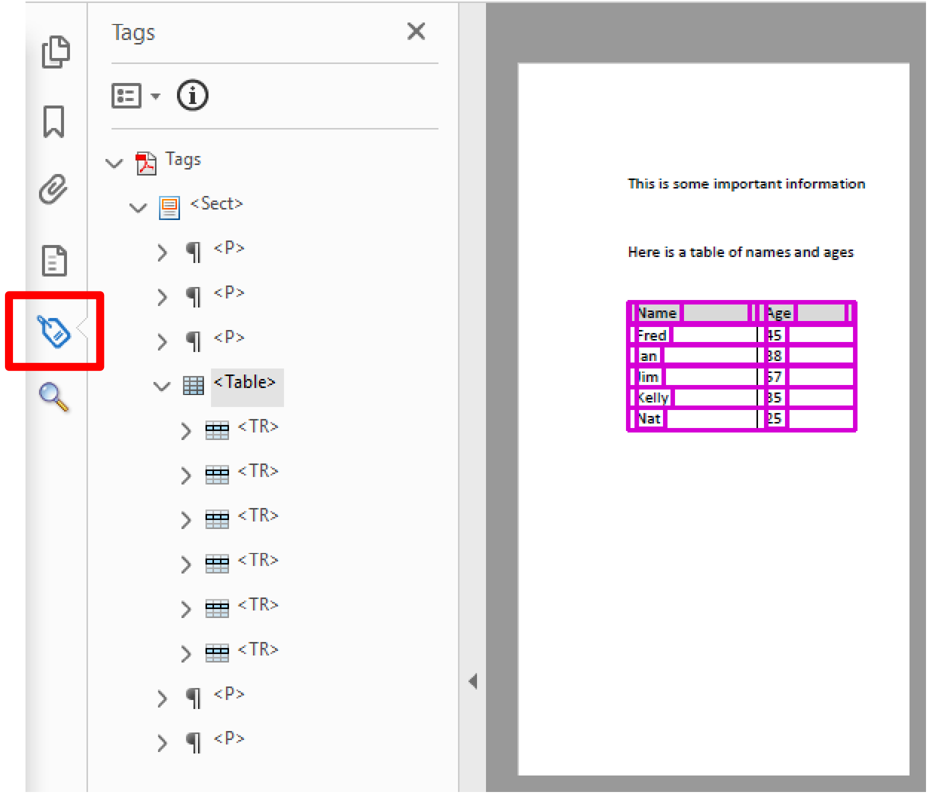


This is some important information

Here is a table of names and ages

Name	Age
Fred	45
Ian	38
Jim	57
Kelly	35
Nat	25

Opening the Tags option shows that tags exist



```
Tags
  <Sect>
    <P>
    <P>
    <P>
    <Table>
      <TR>
      <TR>
      <TR>
      <TR>
      <TR>
      <TR>
    <P>
    <P>
```

This is some important information

Here is a table of names and ages

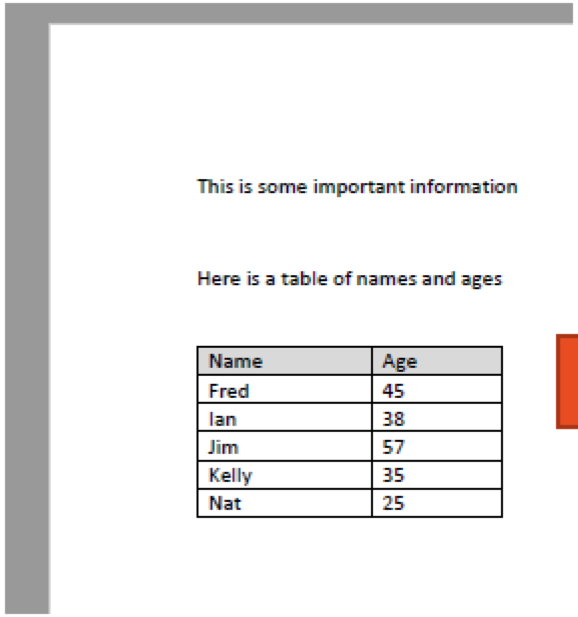
Name	Age
Fred	45
Ian	38
Jim	57
Kelly	35
Nat	25

The entire table is considered to be within a <Table> element

Each row is considered to be within a separate <TR> element

Solid Framework (by default) uses this option when reconstructing the document

PDF

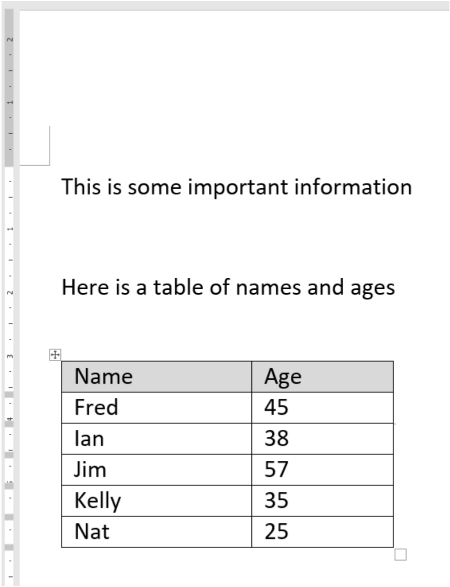


This is some important information

Here is a table of names and ages

Name	Age
Fred	45
Ian	38
Jim	57
Kelly	35
Nat	25

Word



This is some important information

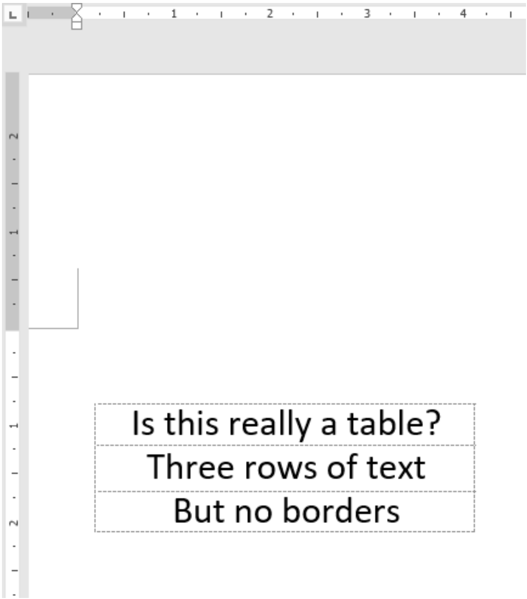
Here is a table of names and ages

Name	Age
Fred	45
Ian	38
Jim	57
Kelly	35
Nat	25

This can give *extremely* accurate reconstruction

So what is the problem?

- Files that look similar can give different results if one is tagged and the other is not.
- The tagging may also be incorrect which can be confusing



Is this really a table?
Three rows of text
But no borders

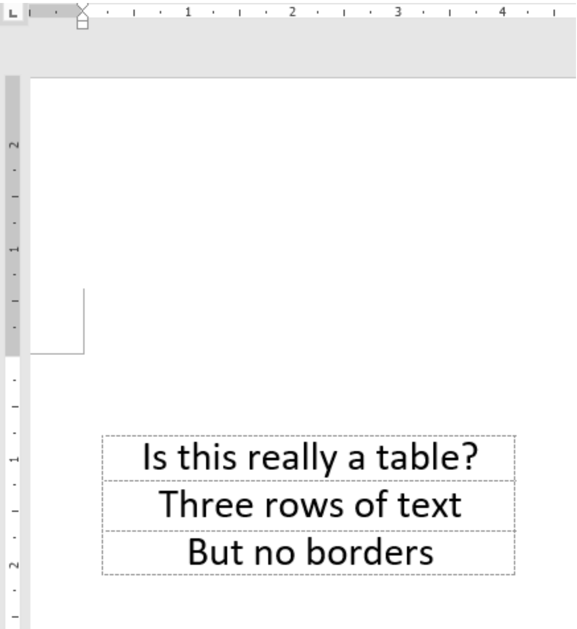
In this sample, the Word document used a table to contain the text

Different files will be produced if the PDF is saved with Tags, without Tags, retagged within Acrobat, or created by using a print driver.

Nonetheless they are all “valid” PDFs.

Effect of tags on Reconstruction of the Word Document

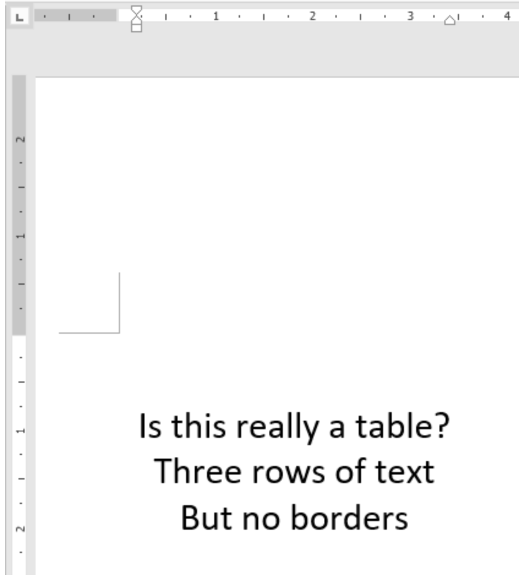
PDF created with tags



Is this really a table?
Three rows of text
But no borders

Reconstruction contains a table

PDF created without tags



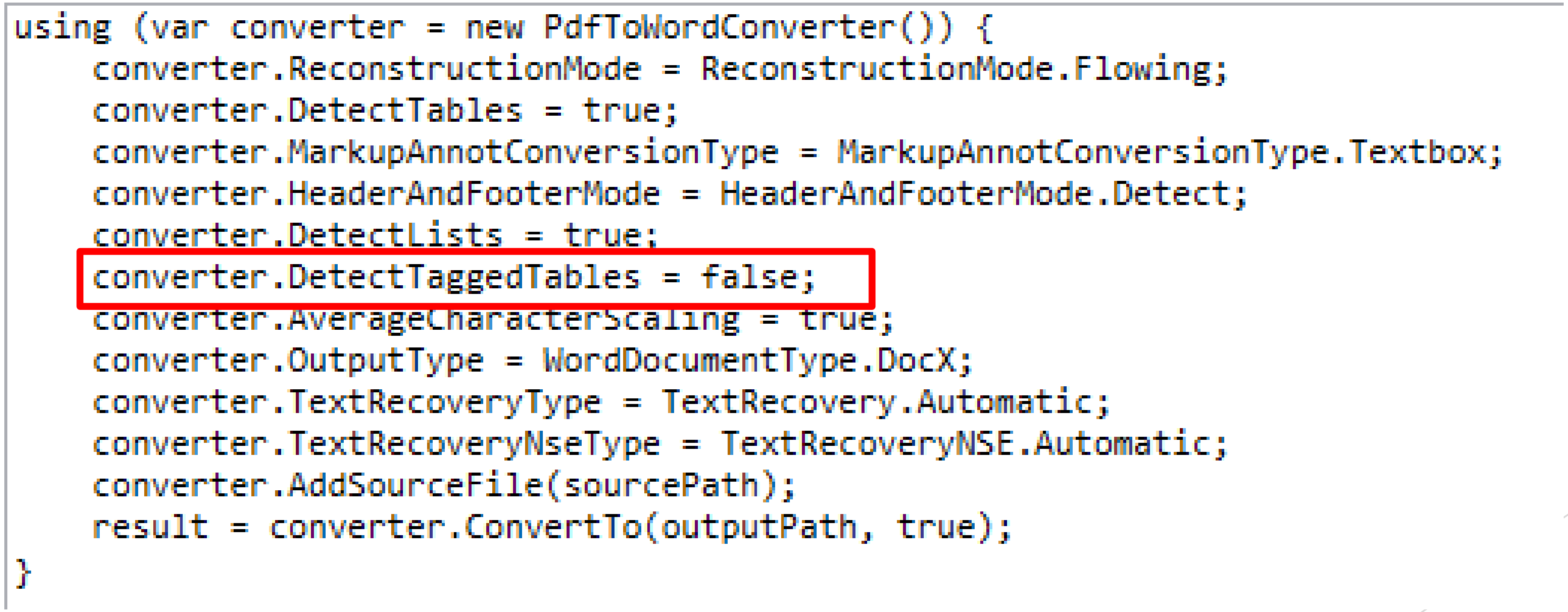
Is this really a table?
Three rows of text
But no borders

Reconstruction does not contain a table (since the data is not really very table-like)

This can be a significant problem if you wish to compare two PDFs and identify the differences between them

Ignoring Tags – Option 1

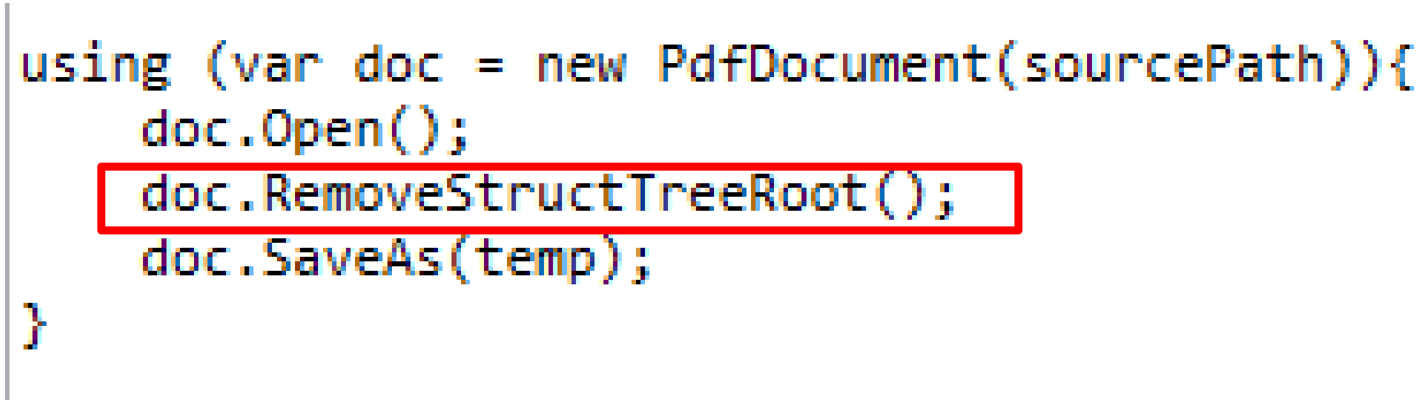
Use the option DetectTaggedTables = false (default value is true)



```
using (var converter = new PdfToWordConverter()) {
    converter.ReconstructionMode = ReconstructionMode.Flowing;
    converter.DetectTables = true;
    converter.MarkupAnnotConversionType = MarkupAnnotConversionType.Textbox;
    converter.HeaderAndFooterMode = HeaderAndFooterMode.Detect;
    converter.DetectLists = true;
    converter.DetectTaggedTables = false;
    converter.AverageCharacterScaling = true;
    converter.OutputType = WordDocumentType.DocX;
    converter.TextRecoveryType = TextRecovery.Automatic;
    converter.TextRecoveryNseType = TextRecoveryNSE.Automatic;
    converter.AddSourceFile(sourcePath);
    result = converter.ConvertTo(outputPath, true);
}
```

Ignoring Tags – Option 2

Create a copy of the PDF without tags using **RemoveStructTreeRoot** before conversion.



```
using (var doc = new PdfDocument(sourcePath)){
    doc.Open();
    doc.RemoveStructTreeRoot();
    doc.SaveAs(temp);
}
```

Try It Yourself

- The files used in this presentation are available so that you can try the conversions yourself. Click on the file names to download the files.

File name	Notes
saved_from_word_with_tags.pdf	Created from the Word document by "Saving as Adobe PDF" with tags enabled
saved_from_word_without_tags.pdf	Created from the Word document by "Saving as Adobe PDF" with tags disabled
retagged_in_acrobat.pdf	saved_from_word_with_tags.pdf opened in Acrobat and automatically retagged. This resulted in the <Table> tag being removed
printed_from_word.pdf	Created from Word using "Microsoft Print to PDF"

Summary

- Tags are generally great and help to give excellent reconstruction
- Sometimes tags can alter the way that text is reconstructed
- Solid Framework has the ability to ignore tags during reconstruction, or even to remove them entirely