Table Frames Interfere with Adjoining Borders

This document is provided to give a 2nd working copy of the same bug as reported in Python3docs Bugs #4, with extra information supplied.

Comment

The report (below) was made at BugZilla#164298 on Dec 12 2024. The bug was first documented at GitHub Python3Docs Bug#04 and then documented separately as Bug#04.

a) Version 24.8.3.2 (Build) is affected by this bug
b) Grandmother version 3.3.0.4 is affected by this bug
c) All other versions between 3.3 & 24.8 are affected by this bug.
d) Michael Stahl declared that *"this is working as designed"*™ (good grief)

I now understand that the reason that LibreOffice has been riddled with bugs for the last 7+ years is *by design* rather than through ignorance, laziness or inability. There now follows extra comment + demonstration of this bug:–

original bug report:

> In Writer, if a paragraph with inline-border is placed directly below a table-inside-a-frame, then the below-frame spacing of the table-in-frame will be added to the internal top-spacing of the next paragraph, rather than being placed between the two entity's borders.

Extra comment:

> The original report is almost accurate (if wordy), though it is probably more accurate to say (as in the H1 Heading at top) that a Table Frame *"Interferes"* with adjoining borders. It also helps if a person can see this in action. These are the features to pay attention to:–
>
> 1. I believe this may be a bug common to all framed entities.
> 2. The bug seems to occur with *anything* that is framed and is positioned next to something with a border.
> 3. The point below is that 3 of the paragraphs have a blue inline-border & thus the bug becomes discernable.
> 4. There are 5 demonstration pages below; the 1st two pages are individual examples of the two ingredients, whilst the final 3 pages are snapshots in a very short film.

How to produce this document

1. This document created from scratch under 24.8.3.2. `menu:Help | About LibreOffice` shows: Build
2. The entire document bug03+.odt copied as this doc & then edited.
3. The table "Table 2.3: Integer Conversion Functions" within chapter_02.odt then copy/pasted into page#5 of this doc as a replacement for the former image+caption + the outer frame changed from a thin blue to a #0A2746 dotted border (same as inner borders). Finally a frame is added (the frame is anchored to a paragraph & set with 0.5 cm bottom spacing, then set to place itself at paragraph centre-top).

   There is 1 paragraph on that page. The text "some words" is placed below the table simply to visually indicate where the Pilcrow ends.

   Paragraph pilcrows can be displayed via either `Ctrl+F10` or by placing a tick into `menu:View | Formatting Marks`.

   Note that there is a longstanding bug in the Navigator (`F5` or `menu:View | Navigator`): whilst editing, the table will accept a rename to "Table 3.1: Integer Conversion Functions", but `Navigator | Rename` will not accept the dot (U+002E) in that name for the same table.
4. 4 paragraphs from p8 of chapter_01.odt were left as-is within page#6. (the *last 3 paragraphs are* Code Box 2 (*CB2*) *paragraphs, which have an inline text-border*)
5. The table on page#5 was copy/pasted as a replacement for the former image+caption in page#7.

   The end result is the words "para1", "para2" at the top of page#7, followed by the table, then "para3", then a plain-text paragraph, and finally a *CB2* 3-line paragraph (with embedded blue border).
6. The next 2 pages (page#8 + page#9) were emptied of all content & the entirety of page#7 was copy/pasted in place. For each of those pages, the framed Table was selected on the page using the Navigator (`F5`) and then the <shift>+down-arrow cursor (↓) was pressed.
7. For page#8 the cursor was pressed until the words "Let's look … the details:" appeared above the image
8. For page#9 the cursor continued to be pressed until the *CB2* top-border appeared above the image. That was much more difficult to do. Framed table 4.3 is 1.01 cm from the top of it's paragraph, but framed table 4.4 is 0.51 cm from the top of *it's* paragraph, and I was only able to finally get it there via editing the `Position | Vertical` setting.

Demonstrations:

Refer to the final pages (page#7 to page#9) for a practical demonstration of this bug.

Things to note:–

i. to recap:

   a) Each framed table in all example pages is identical other than name.

   b) Further, each framed table is set with 0 cm borders; Wrap is OFF; spacing is 0cm except for bottom = 0.5 cm.

   c) All text paragraphs are identical in borders + spacing to the framed table, with the sole extra wrinkle that *Code Box 2* (*CB2*) paragraphs have the check-box selected for *"do not add space between paragraphs of the same style"* (which means that only the bottom paragraph has a bottom spacing).

   d) The combo of the above means that text elements can 'flow' around the frame of the framed table. In addition, it also means that, once selected, the framed table can be moved up & down the ladder of the surrounding text paragraphs by means of the keyboard up-(<shift> + ↑) or down-arrow (<shift> + ↓); in this instance <alt>+arrow is ineffective.

ii. in page#7 *(Table 4.2):*

   a) The bottom spacing below ALL paragraphs + image is 0.5 cm

   b) The 3 x *CB2* paragraphs are surrounded by an embedded blue border & the spacing from text above to the top of the border is 0.5 cm.

   c) Everything looks perfectly normal. Nothing to see here.

iii. in page#8 *(Table 4.3):*

   a) This is the clearest demonstration of the bug.

   b) The text paragraph "Let's look … the details:" is above the framed table whilst the *CB2* paragraphs are immediately below.

   c) The spacing above the table looks a little too small whilst that below looks a little too large. However, none of that is very obvious until you look for the *CB2* top-border…

d) Where has the *CB2* top-border gone? The frame lower-spacing is supposed to be between the bottom of the table & the top of the next paragraph. Instead, that top-border is Missing In Action.

iv. in page#9 *(Table 4.4):*

a) This is probably the most blatant demonstration of the bug.

b) The *CB2* top-border now reappears above the image. I believe this to be a display error, in that the *CB2* border *'belongs'* to the *CB2* text & should not be separated from it. The current display looks daft.

c) This daft bug has been in place for more than 7 years, and is a Grandparent gift from Apache (if I recall the heredity of LibreOffice correctly). Time to fix it, methinks.

v. All frames show this bug.

Examples

(see next pages):

Table 4.1: Integer Conversion Functions

Syntax	*Description*
`bin(i)`	Returns the binary representation of int i as a string, e.g., `bin(1980) == '0b11110111100'`
`hex(i)`	Returns the hexadecimal representation of i as a string, e.g., `hex(1980) == '0x7bc'`
`int(x)`	Converts object x to an integer; raises `ValueError` on failure—or `TypeError` if x's data type does not support integer conversion. If x is a floating-point number it is truncated.
`int(s, base)`	Converts `str s` to an integer; raises `ValueError` on failure. If the optional base argument is given it should be an integer between 2 and 36 inclusive.
`oct(i)`	Returns the octal representation of i as a string, e.g., `oct(1980) == '0o3674'`

some words

Let’s look at a few tiny examples, and then discuss some of the details:

```
x = "blue"
y = "green"
z = x
```

para1

para2

Table 4.2: Integer Conversion Functions

Syntax	*Description*
`bin(i)`	Returns the binary representation of int i as a string, e.g., `bin(1980) == '0b11110111100'`
`hex(i)`	Returns the hexadecimal representation of i as a string, e.g., `hex(1980) == '0x7bc'`
`int(x)`	Converts object x to an integer; raises `ValueError` on failure—or `TypeError` if x's data type does not support integer conversion. If x is a floating-point number it is truncated.
`int(s, base)`	Converts `str s` to an integer; raises `ValueError` on failure. If the optional base argument is given it should be an integer between 2 and 36 inclusive.
`oct(i)`	Returns the octal representation of i as a string, e.g., `oct(1980) == '0o3674'`

para3

Let's look at a few tiny examples, and then discuss some of the details:

```
x = "blue"
y = "green"
z = x
```

para1

para2

para3

Let’s look at a few tiny examples, and then discuss some of the details:

Table 4.3: Integer Conversion Functions

Syntax	*Description*
`bin(i)`	Returns the binary representation of int i as a string, e.g., `bin(1980) == '0b11110111100'`
`hex(i)`	Returns the hexadecimal representation of i as a string, e.g., `hex(1980) == '0x7bc'`
`int(x)`	Converts object x to an integer; raises `ValueError` on failure—or `TypeError` if x’s data type does not support integer conversion. If x is a floating-point number it is truncated.
`int(s, base)`	Converts `str s` to an integer; raises `ValueError` on failure. If the optional base argument is given it should be an integer between 2 and 36 inclusive.
`oct(i)`	Returns the octal representation of i as a string, e.g., `oct(1980) == '0o3674'`

```
x = "blue"
y = "green"
z = x
```

para1

para2

para3

Let’s look at a few tiny examples, and then discuss some of the details:

Table 4.4: Integer Conversion Functions

Syntax	*Description*
`bin(i)`	Returns the binary representation of int i as a string, e.g., `bin(1980) == '0b11110111100'`
`hex(i)`	Returns the hexadecimal representation of i as a string, e.g., `hex(1980) == '0x7bc'`
`int(x)`	Converts object x to an integer; raises `ValueError` on failure—or `TypeError` if x’s data type does not support integer conversion. If x is a floating-point number it is truncated.
`int(s, base)`	Converts `str s` to an integer; raises `ValueError` on failure. If the optional base argument is given it should be an integer between 2 and 36 inclusive.
`oct(i)`	Returns the octal representation of i as a string, e.g., `oct(1980) == '0o3674'`

```
x = "blue"
y = "green"
z = x
```