Quality Without Compromise Since 1969

PubLink
Professional Composition Software

DeskTopPro Release 16.50

DeskTopPro's 16.50 follows our last Release 16.00. This release notice includes all enhancements and fixes completed since that release. Please read this documentation carefully and completely.
Should you have any questions, please give us a call—**410.612.9067.**

HIGHLIGHTS

NEW PRODUCTS

PubLink on PC. PubLink's DeskTopPro software is now available to run on any x86-based server platform running Sun Solaris operating system. Sun Sparc hardware is no longer necessary; any PC platform capable of running Linux server software can now run Solaris and PubLink. This is now a standard platform-migration path, and is available to any PubLink support customer for only a moderate upgrade fee. Please contact PubLink if you are interested.

PLwebservJ. Our new web-based processing server, PLwebservJ, is a web portal for hands-off PubLink document processing. Your web server platform runs our java-based module, which receives processing instructions from a customizable web browser application and invokes the DeskTopPro processes on a Solaris PubLink server. The web application, which runs on local or remote connected browsers, gathers document IDs and processing instructions from operator, submits requests to the server, and receives result messages and, optionally, PDF pages back for display. This is an add-on option for your production system. Please contact PubLink if you are interested.

NEW FEATURES

Hyphenate compound words. PubLink's hyphenation logic now will properly break compound words, by finding their component pieces in your exception-word dictionaries. Especially useful for drug/chemical names and for the German language.

Continued-lines at foot. Continued-lines at top of a column, enabled by special design-text frames in the master layout, may now be also applied at foot of column, for "continued next page" lines where appropriate in the text flow.

Shaded text backgrounds. Apply shaded/tinted backgrounds behind short or long text phrases, in the same way that underscores or strike-through rules are applied. Uses an extended strike-through rule ([sr) command. Easy control of positioning, height, color and shade %.

ENHANCEMENTS OF NOTE

Long graphic names. Any imported graphics may now have a name up to 57 characters long.

Long textfile names. Member text-file names may now be of any length. Long names are resolved by a wrapper function at time of import and (optionally) export. Within the system at menus, at EditMaster and in logging, a shortened but unique form of each long name is used.

Textfile in xml. As an additional method of editing or filtering a text file, it can now be exported from our normal DeskTopPro binary .txt format to xml; then filter, probe or edit it as needed with xslt or other tools; then re-import to binary.

Hyphenation control. New command [nh to inhibit word break at point of command.

Better control of head lines. In pagination, no-break areas based on line count above and below heads now reliably include only visible text lines. Empty or zero-depth lines are ignored in these line counts.

Tinted background boxes that break to next column. For tinted background boxes with rounded corners ([bfr) that break to next column, the borders at break point will now be continuous straight lines instead of rounded at each stop and re-start point.

Publisher Strings. New operators for XML Publisher string variables, allow you to join text strings together.

Skips or Blanks. When blank-page layouts (new in Release 16) are in use, new [bd (Blank Drop) command acts as override to actually skip a page, at a skip point such as [no, instead of setting the blank page.

Archive logs. New option to archive each successive MasterPage log file. Useful for tracking the page count or alert/abort history.

Proofing. Print galleys side by side on pdf pages.

Current Affairs!

Supported or sanctioned product revision or release numbers are:

PubLink Rev.: **16.50**	Solaris: **10**
SGML/XMLPublisher: **3.2**	Ghostscript: **8.15**
Penta Unix (PUX): **sol10**	Hummingbird Exceed: **2009**
DataTool (pdata): **2.04**	Adobe Illustrator: **CS3**
Utils: **16.50**	PDFs: **Level 1.4**
MasterPage Logic Rev.: **11**	**Postscript Level 3** Support for EPS Files
PLWordIn: **1.0**	TCL/tk revision: **8.0**
FileMerlin: **8.0**	

Research & Development!

In order to better serve our customers PubLink is constantly following the industry trends in an effort to lead the industry with new products and procedures made available to our customers. To that end, since our acquisition of the standard-setting 'Penta System' Software in 2002, we have acquired new personnel and customers. Our development program has turned toward opening up the system for easy operation from any workstation, anywhere. That program is bearing results in 2010 and beyond.

One thing that isn't obvious in your new products, enhancements, and fixes is the effort spent planning, evaluating and developing them, so that you can immediately use them productively.

Remember we have lead and outlasted every major typesetting and pagination product and company since the product's inception in 1969. We will continue to offer standard-setting speed and power in document production, innovative products to control the core processing, and the attentive service you need to back them up.

Revision Process!

A software release is referenced and referred to by its release number (first 2 digits), and the version of that release is the number after the decimal (i.e. .03). The structure of PubLink's DeskTopPro software release numbering is as follows.

Feature	Type	Distribution	Reason/Purpose
Release 16.00	Major	All SSA	Release Name
Revision 16.01	Alpha	PubLink	Develop/Assemble
Revision 16.02	Beta	Distributors	Testing
Revision 16.03	Shipping	All SSA Users	General release
Revision 16.50	Enhancements	All SSA Users	General release

Approximately once a year you will receive a major or enhancements release from PubLink. Every year we make software adjustments and data changes which are deemed necessary for our users' production. These items appear in the next release for all SSA customers and have usually been used in the field by the requesting customer prior to shipping. New features and enhancements are tested at beta sites prior to release. Every once in awhile there will be an easily correctable anomaly with a release, such as a permissions problem, but PubLink takes great pride in delivering fully tested software releases with accurate documentation to you, our valued customer.

QUALITYPE / H&J

Enhancement: Hyphenation of compound words

The need: Compound words, ie. words made up of two or more whole words without any connecting punctuation, may or may not hyphenate correctly. To ensure proper hyphenation, these words may be predicted and entered whole into the exception-word dictionary. Otherwise, H&J's probability logic takes over and may result in an incorrect word break. A usage such as ‘‘initially-approved’’, with the hyphen, is not a compound, and breaks properly.

However in certain domains, words may combine in endless variety, making it impractical to predict and enter all into a dictionary. This is a normal situation in the German language (eg. *Eisenbahn* and *Fahrplan* combine to *Eisenbahnfahrplan*), and in drug and chemical names (eg. *tetrahydroxytrialuminum*). Especially in these domains, the program should recognize the component words of a compound, and hyphenate it correctly.

Solution: PubLink's hyphenation logic will now properly break compound words, when the component words are found in the dictionary database. It will break either at a join point between the two (or more) dictionary words, or at any valid breakpoint in one of them. It does this by first searching your exception-word dictionaries for the entire long word; if not found, it matches the longest initial sub-word. It records legal breakpoints within the matched sub-word and one at its end, then repeats the process for the remainder of the long word, until it is fully matched. If a piece is not found in the dictionaries, the long word is broken using the fall-back probability logic.

This repetetive matching process will, in fact, also nicely handle word prefixes. You could enter a prefix like *anti* into your dictionary, as well as the combining words *oxidant*, *establishment*, *war* and *hero*, with the result that the four combined versions (*antioxidant* etc.) would all break correctly at the join point, without the need to enter each into the dictionary.

Enhancement: New hyphenation no-break tool

Description of problem: There are times when it is useful to inhibit a word break at a certain point in a word or meta-word. For instance, for a word broken incorrectly by the hyphenation logic, that is not yet in the exception-word dictionary. Or, if the meta-word-break logic (new feature in Rev 16.00; see Release Notes) chooses to break a meta-word in front of an inappropriate in-text equation element.

Solution: Insert new command [nh (no hyphen) at a potential break point, to prevent the hyphenation program from breaking the word there. It will instead choose a different allowable breakpoint. This command would normally be used as an editing tool to correct bad word breaks.

Fix: Apply AJ page-fit leading only to real lines

Description of problem: Using the [aj page-fitting tool (new in Rev 16.03) to adjust line leading over a range of text, user got undesired result: A horizontal rule forming top edge of a photograph, was separated from the photo by a small gap of white, the depth of the [aj adjustment.

Solution: H&J has been modified so that [aj modification of body leading (`[aj,l2]`) will have no effect on zero-lead lines.

Fix: Rev16 metaword hyphenation caused numbers to hyphenate

Description of problem: The meta-word hyphenation enhancement contained in Rev 16.03 came with instructions to make some changes to the ‘‘H&J Recode Tables’’ data (file userdata/postscript/layouts). Specifically, these changes were the X and Y ‘‘recode’’ values of the comma and the ten digits. Unfortunately, those changes not only enabled hyphenation of numbers, as needed by the meta-word logic, they also started numbers breaking in the traditional hyphenation logic, resulting in some lines in standing work breaking differently.

Solution: Workaround in all cases was to undo those data changes. Users that wanted to use both flavors of hyphenation logic had to maintain two sets of data, and enable one or the other for different jobs.

Now, H&J has been fixed so that your comma and digit recodes can (and should) be restored to your original values, and traditional hyph logic will break words and numbers as before. The meta-word logic now no longer depends on changing these items in the recode table.

Required data changes: If you changed no system data for metaword hyphenation after installing rev 16.03, do nothing now. Otherwise, in pdata (DataTool), choose "H&J Recode Tables". Open recode table 0. Then click "Next Pg" for the second page.

Examine:

- row 44, the comma. The Zero Shift column should be set to its values before Rev 16.03, normally 7/0/28.
- the ten rows 48 through 57 containing recodes for the ten digits, where the Loc column contains 65–74. The X & Y values should be set to as before Rev 16.03, normally 9/0.

Click ’Save’ if you have made any changes, and Exit.

Fix: Inconsistent line break at leader fill

Description of problem: For a leader-fill line that exceeds measure and must break over to two lines, there are seven break styles to choose from: Leader dots before break, after break, both sides of break, one word plus leaders after break, etc. However, these all reverted to just one style IF there were two or more words before the leader-fill command.

Solution: A bug in the logic of breaking of leader-filled lines has been fixed.

Fix: H&J could crash at random

Description of problem: The H&J program could freeze or crash with no indentifiable cause, other than having processed long or complex documents, or many documents.

Solution: A memory leak was tracked down in H&J program, which could result in memory overflow, causing crash. We predict this fix will greatly reduce the number of occasional, unexplained H&J failures.

Enhancement: Continued lines at foot of column

Description of need: MasterPage offers the continued-lines feature for directory or dictionary publications, allowing up to nine levels of subject lines to be repeated in sequential heads at the top of each new column or page on which they continue. For example, column one of a directory might start with ‘‘Private Clinics - *continued*’’, and underneath that: ‘‘The Miller, Johnson and Jones Hearing Clinic (cont.)’’, followed by the continuing column. However, there is no facility to have a corresponding continuation line at the foot of a page or column, indicating that the subject continues next page and is not yet done.

Solution: MasterPage now features continued-line foot frames, that operate by the same principle and method as the heads. User adds foot frames to the bottom of each column of the even and odd master layouts where they are needed, and flags them as design text of type continued-line. In building pages, MP will activate a frame if the text-related string it calls out is in effect on the last line of its column *and* in the first line at top of next column. If such is not the case, the frame and its depth will be suppressed. CL foot lines are more limited than heads, in keeping with the more limited need for them: The program supports only one level of continued-line foot line, and it does not send a frame’s text-related content to a pre-H&J to determine how many lines the foot will fill. Instead, user designs a foot frame to a fixed depth, probably one or two lines.

How to use: In the master layout, draw a design-text frame sitting at the foot of each flow-text column where you will want continued-lines to appear when called for. The frame needs these values:

Lockpoint=**BL** (bottom left)
Horiz=(that of the text col)
Vert=***SA1** (same as 1), where ‘1’ is the flow-text column’s frame number.
Width=(that of the text col)
Depth=your choice
Overlay: Click ***on***
Relative frame#=0
Text/CL/SN/AR: Click ***CL***
Text box at bottom: Exactly as in a continued-line head frame, enter style commands and text that should make up the continued-line, plus a callout to the text-related string that will be tested to see if it is active in the text at bottom of column. Callout consists of *stringname*, *comma*, ‘‘*last*’’ within braces. Example:
[fy50,1,10,12]{clinicname, last} (cont. next page)[ep

Your articles should contain TR-strings defined with the [xt and [xx commands, as:
`[xtclinicname,The Miller, Johnson and Jones Hearing Clinic[xx`.

Enhancement: Specify continued-line frames in any layout

Description of problem: To activate continued-line head lines in a unit, the special frames to hold them must be defined in the unit's default even-page master layout. If that page layout has multiple columns, the left-most column must hold them. Cont-line frames may then also be specified at the top of other columns, and in the default odd-page layout and in other layouts used by the unit (including those called by the [nl command). But it's the CL frames defined in column one of the default even-page master that specify the text-related string names (implemented by [xt*name, string content*[xx in the unit's text), and their heirarchical levels one to nine, that will be used throughout the unit. Frames defined in other layouts merely specify how many levels are used in which columns, and the text and style to be used in each. This requirement exists because MasterPage assumes, with reason, that any publication having continued-lines at tops of columns will always need them, at minimum, at the turn of page, ie. top left, and may also have them at start of the right page and at tops of other columns.

However this requirement proves too restrictive, when [nl calls different layouts later in the unit, and when those layouts need continued-lines based on a different set of text-related strings; or when continued-lines aren't used at all at start of unit, but rather commence at a later [nl point.

Solution: MasterPage has been upgraded to remove this restriction. Continued-lines may now be started or changed in any layout, including the default odd, the chapter opener, a special layout assigned to a specific page, or one called by [nl. The text-related string names named in the new layout will be honored. This enhancement also works in conjunction with Rev 16.50's new continued-line foot line feature.

Enhancement: Improve pagination with predictable line counts

Description of problem: MasterPage bars breaking a page within a count of lines in certain situations. These no-break situations are: Line count above a head (regular [bh or special [sh3,2] head), line count below a head, line count on last page, and line count before a forced new page ([np). These count-based no-break areas prevent a page from eg. starting with just one line before a new head, ending with only one line below a head, or consisting of only two lines alone (before next chapter on next page). User provides these line counts in the Breaking Style, Chapter Style, and [sh command. However up until now the number of lines has been a tricky quantity, because the software counts *all* lines, including invisible test lines, zero-lead lines, and empty lines. These non-'real' lines exist for different purposes, including just extra space, but when they fall in a zone being counted for non-breaking, the program counts them when it really should not. As a result, for instance, a page can end up with just two visible text lines at top before the head, when your rule is three lines.

Solution: H&J has been enhanced to analyze and flag each line as being either real, ie. containing real, visible text or horizontal rules, or not real, ie. containing no visible marks. MasterPage then bases all its line counting on real lines only. Note that each table line, that begins with [bt and ends with [et, is considered one visible line, for counting. This is true even if cells within the tab line contain multiple lines of text. A table line is also always considered as ending with [ep, for purposes of avoiding orphans.

This enhancement is active *only* for jobs created under Rev 16.50 or later, or bearing LogicRev 11. You may use the shell script 'logicrev' to probe and change the LogicRev # of any job.

Enhancement: Improved results with New Layout command

Description of problem: Using the New Layout command ([nl) as a tool to switch frequently between single-column text, double-column text, and tabular material, in conjunction with [bm2] / [em straddling areas, is an application for which the [nl was not initially designed. Multiple short [nl and/or [bm2] sections on the same page can cause unexpected breakdowns in the page make-up.

Solution: In several ways, the [nl command has been tightened up to better handle interaction with other [nl sections and with [bm2] sections:

- NL can now immediately follow a [bm2] / [em head.
- NL and BM2 can be on same page without risking a page left short.
- Handle end of section, before another NL, better, and diffently than end of page.

New Tool: logicrev script

Description of problem: Each new PubLink software revision features improvements to the pagination logic, along with an associated ‘‘logic-rev’’ number. These improvements are activated only for units that are flagged with the new logic rev number, so that pre-existing paginated work (flagged with an earlier number) will not change their page breaks if re-run. Units created under the new software automatically get the latest number, and earlier jobs may be manually flagged with the new number, to benefit from the enhancements. However, it has been cumbersome and tricky to probe a job for its number, and to change it.

Solution: Rev 16.50 includes a command-line tool called ’logicrev’ that easily and securely lets you find out the logic rev number of any job, and change it. It reports on the named project’s parent or child projects too, for consistent project management.

How To Use: In any shell window, enter ’`logicrev`’ with no argument. This returns full help on command usage.

Enhancement: Keep MasterPage log-file history

Description of problem: User wants to track summary pagination results and production history of earlier pagination runs of a project, to see how edits affect the page count, illustration placement, page alerts/aborts, and other data shown in the MasterPage.log file

Solution: The system will now, optionally, keep all older MasterPage.log files back to a given age, allowing user to compare them.

How To Use: Please see separate document *r16.06.docMPloghist.pdf*, delivered with this release, for full description of this enhancement.

Associated data changes: (See above document)

Enhancement: Repeat heads for in-line rotated tables

Description of problem: The Illustration Style's Breaking Deep Tables screen offers style choices for how MasterPage should handle rotated tables that break to next page. All Illustration Style items, including these for deep tables, describe how to treat *floating* illustrations and tables. Tables keyed in the flow of text, on the other hand, not bounded by [ts and [tx, are positioned where they fall, and do not use these style items. However, the Deep Tables item ''Repeat boxhead on right page for rotated table'' would be helpful for an in-line table that is rotated, by being set in a special rotated master layout. In such tables, every continuation page, left and right, gets a repeatable box head if one is defined. It may not be desired on the right page.

Solution: MasterPage has been enhanced so that the Deep Tables option shown above applies to in-line as well as floating tables. It is highly unlikely that a publication will have *both* types of rotated long tables, each of which is to have a different table-head style on the right-hand page, so the one option governing both table types should be adequate.

Fix: Repeated table head caused misplaced footnotes

Description of problem: When a running table breaks to the next page or column, with a repeated box head placed at top, and when the depth of that table head differs from the depth of an earlier table head, then footnotes at bottom of page can be misplaced vertically, oversetting the text or other footnotes.

Solution: MasterPage has been fixed to account for any changes in the depth of repeatable table heads, so that they will not affect placement of other page elements.

Fix: Blank pages failed with facing-page alignment

Description of problem: Using the blank-page layouts feature introduced in Rev 16.03, user found that pagination fails, on a page containing [no which should generate a blank page. This happened only in a page pair that was aligned short or long by the facing-page alignment logic.

Solution: Blank pages did not work correctly with facing-page alignment. This has been fixed.

Fix: Text could overset footnote

Description of problem: On a page containing both type-2 footnotes (that is, footnotes in columns independent of the text columns) and floating illustrations or tables, where a footnote is called out on last text line of page, the footnote could be overset by a text column.

Solution: This poor page result was found to happen when the page ends with a "borderline" footnote callout that might be pushed to the next page. It is caused by a bad interaction between the type-2 footnote placement-retry logic, and the floating-illustration placement-retry logic. That faulty interaction has been eliminated.

Fix: Footnotes swapped between pages

Description of problem: Occasionally, in SGML/XML Publisher units, footnotes were output on the wrong pages. This happened when one or several, but not all, of the unit's member text files were re-imported from xml, after initial import and pagination. It also only happened when H&J separates floating elements such as footnotes from the source articles into parallel _float.txt files.

Solution: Upon re-import of a member article, MasterPage was found to re-record the _float files into its list of members, in a different order. This led to eg. footnotes being drawn from the _float files in the wrong order too. MP has now been changed to guarantee that its members list is always in the correct order, eliminating the problem.

Fix: Floating figures and tables together could corrupt a page

Description of problem: Use of floating illustrations in same chapter as floating tables sometimes caused corrupted page frames. Symptom: Illustration placed below page, pp program crashes when try to print a page.

Solution: Although floating illustrations and tables are processed in the same way by MasterPage, tables carry extra data. In one circumstance, illustrations mistakenly tried to use data meant for tables, resulting in corrupted pointers to page elements, and the symptoms described above. This mistake has been corrected.

Fix: Side-notes don't appear

Description of problem: When a line has two or more sidenotes attached, only the first one was displayed.

Solution: It is unusual to call out two sidenotes on the same line, so the problem was not reported before. Problem now fixed: If a text line calls out two or more sidenotes, all of them will be set sequentially in the margin.

Fix: Problem with consecutive new-layout commands

Description of problem: A new-layout command ([nl) followed just a few lines later by another [nl, could cause the contents of the second section to set all wrong.

Solution: MasterPage uses the user style value ''Minimum lines: Below Head'' to determine the shortest allowable section of a page. A [nl, new layout, command starts a new section. If another [nl is then encountered *before* the user's minimum-line count, the program did not expect this and went a bit haywire. Program has been fixed to handle this situation correctly.

Fix: No-break area overflowed page

Description of problem: In a multi-column publication, a straddling area ([bm2]) that begins mid-page and is flagged using [bn to not break, could overset the bottom of page with no warning message to the console or log file. The overset should at least be flagged with a message, or preferably the whole area carried to the next page.

Solution: Program will now carry such an area to the next page, as the user of the [nb command intends. This change in page result will take effect *only* for units flagged with Rev 16.50's LogicRev 11. Exception: As always, if the no-break area starts at top of page, and runs deeper than the page depth, it stays on the page with an error message.

Required data change: Only units carrying LogicRev value 11 or higher will benefit from this fix. To apply that value:

● New units created under rev 16.50 will have LogicRev 11.

● To bump the LogicRev value in existing units, use the 'logicrev' program, described earlier in these release notes, to find out and then change the unit's number. Enter 'logicrev' in a shell window to get full explanation.

Fix: Inside/Outside override in illustration placement

Description of problem: To place floating illustrations/tables, the [ls and [ts commands offer the Inside and Outside placement overrides, which will force the given cut to be placed only in the inside or outside column on a multi-column page. However, if this instruction is given to a figure or table that is itself two (or more) columns wide, the cut will not be placed at all. Using the override on wide cuts is legitimate for two-column figures on a three-column page, or for tagged documents where the table width is not known at first, but is calculated by a stylesheet or pre-process, and all tables of that class are to use ''Outside''.

Solution: MasterPage has been adjusted to allow multi-column illustrations and tables to use the Inside or Outside placement overrides in commands [ls and [tx.

Fix: Adjustment of extra lead on top line

Description of problem: Extra leading that is expandable but not droppable (eg. [el6,1]) that lands above top line of a column, should be among the extra leads adjusted by the Vertical Spacing table, but it is not. The effect of this flaw is normally minor, but shows up in at least two ways: Side-notes on the page are slightly offset from their callout text lines; in a column that is split into two frames, the frames can slightly overlap.

Solution: The above anomaly has been fixed: Expandable extra lead on the top line of a column is now adjusted along with other extra leads, when the page is compressed or expanded by the try tables.

Fix: Footnotes misplaced on a rotated page

Description of problem: In landscape pages built using a 90 ° rotated master layout, regular (type-1) footnotes and the cutoff rule are misplaced.

Solution: Footnotes on rotated pages are now positioned correctly.

DESIGNMASTER

Fix: **Strike-through rules displayed inaccurately**

Description of problem: User does legal work including proofs showing insertions and deletions on the printed page. The strike-through rule (command `[sr-]`*`inserted material`*`[sz`) is used to indicate deletions. This rule prints accurately, but the designmaster wysiwyg display is unreliable in three ways:
1. In a strike-thru that runs multiple lines, only the last piece displays.
2. When the text line has a left indent, the rule starts at left margin.
3. The rule shows as hairline thickness, hard to see.

Solution: All three display anomalies in designmaster have been rectified.

Fix: **Strike-through rules still displayed poorly**

Description of problem: Strike-through rules drawn with the [sr command appeared fine on printed page, but displayed poorly in designmaster wysiwyg:

1. Rule types rotated randomly around displayed page.
2. In tab cells, most rules did not display at all.
3. Those that did always showed up in column 1.

Solution: All these display anomalies for [sr have been corrected in the designmaster page display.

GRAPHICS / COLOR

Enhancement: Support long graphic file names

Description of problem: Graphic image files received from outside departments increasingly have very long names. For processing through the PubLink system, a graphic name was limited to 24 characters. This is inadequate, and forces a rename of files before processing; tracking a lot of graphic files then becomes burdensome.

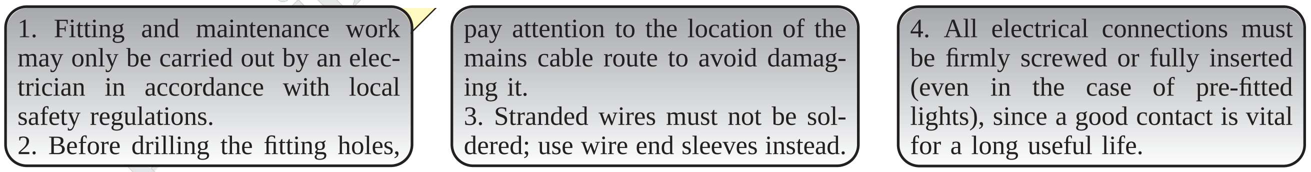
Solution: Filename length for graphics has been increased, throughout the system, to a maximum of 57 characters. This leaves room for the ‘‘.pixel’’ appended by Import, for a total of 63 characters.

Enhancement: Extended control and styles at tinted-box break points

Description of problem: Command [bf draws a tinted background behind an area of text, the depth either fixed or anchored to text start/stop points. In conjunction with command [mc, the tinted area will be outlined by a rule of given thickness. Variant [bfr gives rounded corners to the box or tint area. Commands fields allow control of the area’s appearance: Margins above and below text, margins left and right, color and shade of background and of foreground text, blend characteristics, and thickness and color of border rule. Furthermore, if the depth is anchored to text start/stop points, it will break to next column along with the text. However, user lacks control of such break points: The box outline is drawn complete around each piece of a box that breaks, including the rounded corners (if specified). The result resembles two separate boxes, instead of one continuous. For better results, user needs options for square corners and suppressed outlines at break points.

Solution: The [bf command has been extended with three new [bfr variants: [bfb, [bff and [bfx.

BFR: The current [bf or [bfr command fully ends and restarts the area outline, as if each piece is a separate box:

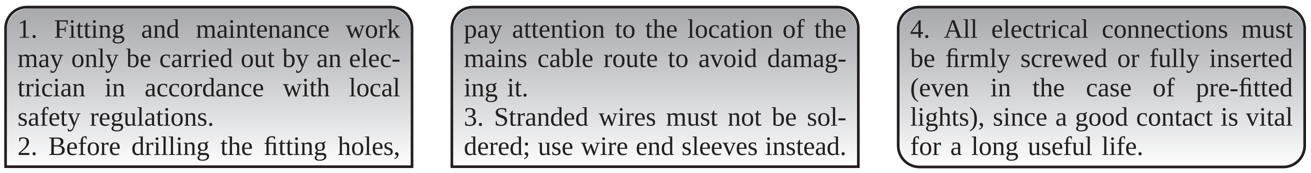


1. Fitting and maintenance work may only be carried out by an electrician in accordance with local safety regulations.
2. Before drilling the fitting holes,

pay attention to the location of the mains cable route to avoid damaging it.
3. Stranded wires must not be soldered; use wire end sleeves instead.

4. All electrical connections must be firmly screwed or fully inserted (even in the case of pre-fitted lights), since a good contact is vital for a long useful life.

BFB: Variant [bfb accepts all same arguments as [bfr, but rounded corners are eliminated before each break:



1. Fitting and maintenance work may only be carried out by an electrician in accordance with local safety regulations.
2. Before drilling the fitting holes,

pay attention to the location of the mains cable route to avoid damaging it.
3. Stranded wires must not be soldered; use wire end sleeves instead.

4. All electrical connections must be firmly screwed or fully inserted (even in the case of pre-fitted lights), since a good contact is vital for a long useful life.

BFF: Variant [bff also eliminates rounded corners *after* each break, so that all break start/stop lines have square corners:

1. Fitting and maintenance work may only be carried out by an electrician in accordance with local safety regulations.
2. Before drilling the fitting holes, pay attention to the location of the mains cable route to avoid damaging it.
3. Stranded wires must not be soldered; use wire end sleeves instead.
4. All electrical connections must be firmly screwed or fully inserted (even in the case of pre-fitted lights), since a good contact is vital for a long useful life.

BFX: Finally, variant [bfx also eliminates the horizontal outlines above and below the breaks:

1. Fitting and maintenance work may only be carried out by an electrician in accordance with local safety regulations.
2. Before drilling the fitting holes, pay attention to the location of the mains cable route to avoid damaging it.
3. Stranded wires must not be soldered; use wire end sleeves instead.
4. All electrical connections must be firmly screwed or fully inserted (even in the case of pre-fitted lights), since a good contact is vital for a long useful life.

To get a square-cornered box without horizontal outlines at break points, use [bfx with a corner radius of zero: [bfx0, ...].

NB: To get any box to break, you must turn on Post-processing in the unit's Chapter Style menu!

Enhancement: **Draw shaded/colored backgrounds to text**

Description of need: Document calls for certain text phrases to be set on a shaded or colored background.

Solution: The Strike-through Rule command ([sr) has been extended to support setting the ‘‘rule’’ under the black type instead of on top; that is, pieces of the line are knocked out by the characters. Combined with the command's current ability to set a rule of any width, color and shade percentage, this enhancement sets rules as in this phrase, as a background tinted field, on which the type sets.

How to use: To set a strike-through rule in the background instead of foreground, simply append ‘‘,1’’ as field seven to the [sr command. If fields five and six (trapping) are unneeded, include them as zeroes. Summary of command's seven fields:

```
[sr_,_,_,_%_,_,_%_,_]
```

Field 1: base line deflection of rule start, in points downward.
Field 2: base line deflection of rule end.
Field 3: weight of rule, in 1/10 points.
Field 4 (optional): color % shade of rule.
Field 5 (optional): weight of trap.
Field 6 (optional): color % shade of trap.
Field 7 (optional): 1=layer the rule behind text.

Enhancement: **TIFF graphics with no preview**

Description of need: Customer who proofs pages from displayed PDFs instead of designmaster wysiwyg wants to be able to import tiff graphics without creating a preview image. They have no need for the preview, and want fastest possible processing.

Solution: Added new option ‘‘tiff-nopx’’to the graphics import utility.

Fix: **Could not blend to 0% (white)**

Description of problem: The [bf command will produce a shaded colored background that will blend a pantone color anywhere in the range 100% to 1%. But, starting or ending the blend at 0% failed with a crash of pp program.

Solution: The error is now fixed. A [bf field of #%0 (where # is any color number) is now a valid field.

Fix: **EPS import, file housekeeping**

Description of problem: When importing EPS graphics that live on a remote NFS file system, the import procedure leaves behind a copy called *graphicname*.orig. These are large and cause archiving issues.

Solution: The graphics import script was found to lack some needed error checking on its copy/rename of graphics to and from temporary versions, resulting in a temp file being left behind. The script has been enhanced to check and handle certain error situations, so that temp files are cleaned up.

OUTPUT / PDF

Enhancement: **Side-by-side galley proofs**

Description of problem: Printing galley proofs results in all galleys end-to-end on the paper. Especially for narrow-measure galleys, this wastes paper. Also for proofreading ease, user prefers that the galleys be printed two per page, side by side.

Solution: A new output-control option allows choice of single or two-up galleys per printed page. The option is available in the shell 'pp' command, but not at the Print Menu. Thus you may access this option by setting up certain print styles to use the option, or by adding it to scripts that perform automated background processing.

To print galleys side by side, add the keyword/value pair ''`Multi 1`'' to your pp command. To print the normal single galley per page, use ''`Multi 0`'' or just leave out the new keyword and value.

Fix: **Footnote disappeared on printed page**

Description of problem: In a parent-and-child project arrangement, if the parent project contains its own text files to be paginated, then after several pagination runs, that unit in the parent can get to a state in which its footnotes appear blank on the ouput pages.

Solution: This problem was due to some confusion in the parent's list of members, between names of text files and names of child units. When the list contains both, they can appear out of order to some programs. PP program has been fixed to reliably find text-file names, so that footnotes now set correctly.

Fix: **Tiny gaps in vertical rule**

Description of problem: On a page with vertical rules down the side of a column or boxed area, tiny gaps in the v-rule could appear when MasterPage expands the column to fit depth of page.

Solution: Gaps appear only at a point where the text column has a zero-lead line (`[ol0]`), and where the column has been expanded by feathering of lines. Problem found and fixed, gaps no longer appear.

Fix: **Table head mis-positioned on turned tables**

Description of problem: For long tables that break page to page on rotated pages, ie. where the master layouts themselves have rotated text frames, the repeated table heads on odd pages (only) were offset from their correct position.

Solution: This particular method of setting long tables was not really anticipated. Program has been fixed to position those rotated table heads correctly on odd page.

Fix: **Core dump of pp on leader fill**

Description of problem: The pp program could core dump, on a line with leader-fill type 6 or 7 and no space bands.

Solution: Found and fixed.

IMPORT

New Feature: Display or edit text file as xml

Description: Rev 16 introduced a converter program ‘‘txcontx2’’ to convert any of your .txt text files, stored in a proprietary binary format, into simple-ascii xml ‘‘.tx2’’ files that still contain all the line numbers, line breaks, composition characteristics, error messages and other pass-through data that EditMaster needs. This xml version is useful for examining and probing the contents of a textfile, and for filtering it into other applications.

Rev 16.50 offers the complementary tx2-to-txt converter, meaning you can now export paginated articles to the tx2 xml format, filter in content or otherwise edit them using an xml or other editor, and import them back into the page unit. This can be a key tool to do things like batch modifications, or modification of indexing tags after a post-process of the first-pass paginated unit.

As part of adding the tx2-in function to the `import` program, a serious bug was uncovered and fixed in import. This bug occasionally caused any file being imported to be truncated, ending up with a short .txt.

16.50 also contains a revised txcontx2 program with minor updates.

How to use: From a shell window, use shell script `txcontx2` (introduced rev 16) to build an ascii .tx2 file from an existing .txt. Then use new shell script `tx2contx` to send the .tx2 file through the import program back into its .txt form in the tree/project. If the .txt already exists (normal), the older version is saved off with the suffix `.pre-tx2in`. To use either script, simply enter its name (`txcontx2` or `tx2contx`) in a shell window, with no arguments, for full instructions.

New Feature: Accept article filenames of any length

Description of problem: The maximum filename length for text files in PubLink’s software is 24 characters. When ascii or xml files with longer names are received from the client, they must be renamed to meet the length limit. This is an annoyance and leads to production tracking difficulties. Also, if on delivery the modified textfiles are to be exported and returned to the client, they must be changed back to their original names.

Solution: PubLink now provides a wrapper function to perform renaming in both directions. The function is tuned to XML Publisher’s import/export strategy. The "trunc" program (for truncate) takes a filename template, checks all matching files in the directory, and renames those whose names exceed 24 characters, in a consistent way. All renames are recorded in a log file in a sub-folder called "truncated_⟨date/time⟩". If export is needed when the job is delivered, you export to the same date-stamped sub-folder, then run the "untrunc" program on it. It will rename all exported files back to their original long names.

How to use: Please see separate instruction sheet "Utility: Handle Long Filenames" (filename *HowtoUse.trunc.untrunc.pdf*,) included with this release.

Fix: Message on re-import

Description of problem: When a textfile is re-imported, after changing the ascii source file, and a copy of the original source file still sits in the import directory, then the import program puts this confusing prompt message to the console:

```
mv: filename-asc: override protection 755 (yes/no)?
```

This message only means that the original file still exists, and asks if it should use the updated source file instead, but that meaning is not clear.

Solution: In the import script, the message and question have been replaced by a clearer message ‘‘`Deleting older filename ascii file: ... Done`’’. Only if the older file has read-only permissions, the script will also prompt you to verify the delete by asking:

```
rm: filename: Override protection (yes/no)?
```

You should always respond ’y’ to this question.

DATA / UTILS

Fix: **Dictionary_update problems**

Description of problems:

1. Updating a hyphenation exception-word dictionary from a word list, using the `dictionary-_update` program, could corrupt the existing dictionary if the word list is empty.
2. Creating a new dictionary left its in-use flag turned on.

Solution: Problems 1 and 2 have been fixed. Also, shell command `dictionary_update` with no arguments now gives you full help on how to use the command to update dictionaries.

AUTOTAB

Fix: **Character alignment failed in table head**

Description of problem: A repeatable table head, defined within an AutoTab table by surrounding the head with [pm1] and [pm2], does not obey the character alignment points in a column, set up with [ta align-here or align-at-character commands.

Solution: Modified AutoTab so that a repeatable head's columns still have no effect on *determining* the character alignment points throughout that column in the table, but the head's columns will now align at the chosen alignment points.

Fix: **AutoTab gutter v-rules wrong weight or missing**

Description of problem: Gutter vertical rules in AutoTab, defined in the [tw command with a v# field, come out wrong weight, or even missing, on printed page. For example: v5 gave no rule at all, v8 gave a rule of weight 6/10 point, v10 gave a proper 1-pt rule, v20 gave a 3-pt rule. Gutter rules in standard tables come out correct, as do rules using the [bx command.

Solution: AutoTab has been corrected to correctly set vertical rules specified in the [tw command.

Fix: **Long non-tab line in a table gave assertion error**

Description of problem: Inside an AutoTab table, a text line (outside of [bt / [et) that would fill more than 120 picas before its [ep caused Assertion Error and core dump of H&J.

Solution: AutoTab's logic flaw in dealing with lines over 120p has been found and fixed.

Fix: **Negative [hm at start of tab cell caused trouble**

Description of problem: In an AutoTab table cell, starting it with [hm of a negative amount, to move first character to left of margin, caused text in later cells that have alignment points to offset. The offset grew progressively later in the table. This offset error seems to happen only in columns that are created with the the [tw command's 'cdw' descriptor.

Solution: AutoTab has a special internal variable to deal with negative spacing at start of cell, but there were two bugs in handling this variable, causing the offsets. Both bugs now fixed, and [hm-x] at start of cell now works properly with no unexpected side effects.

Fix: **Character alignment specification**

Description of problem: In the AutoTab [tw command, you can use the ‘d’ field to specify a sequential list of characters to be aligned on in that column. For instance the field ‘‘… ,d$., … .’’ specifies two alignment characters: First ‘$’ then period. The field accepts up to nine characters, for nine different alignment points in a single column. The ‘,’ and ‘]’ characters normally end any command field, but they may be included in the list by escaping them with backslash. For instance the field ‘‘… ,d$\,., …’’ specifies ‘$’ then comma then period. However, using such an escaped character in the list also ended the field, in error. So in the last example, the column would get only dollar and comma as alignment characters, and would not add period to the list.

Solution: This flaw has been fixed. Escaped comma or bracket no longer end the list of alignment characters prematurely.

XML PUBLISHER

Enhancement: Append one string variable to another

Description of problem: AutoTag global variables in a Publisher stylesheet may hold numbers or text strings. There is a family of operators to manipulate numeric variables (+, −, *, /, %), but nothing comparable for string variables. The ability to combine two strings into a new variable, for instance, as H&J variables can do, would be very useful.

Solution: Publisher's AutoTag instruction set has been enhanced with the new string-concatenation operator ‘‘:=’’. With it, you may append a string in several forms to any existing global string variable. Instruction takes the form: {$var := X}, where X can be another global variable, an attribute (within an attribute code window), a literal string, or the captured-text buffer. Examples:

Instruction	What it does
`{$mytext = "The quick fox "}`	(establish a string variable)
`{$mytext := "jumped"}`	Appends text. $mytext now contains "`The quick fox jumped`".
`{$mytext := $0}`	In attribute code window, appends attribute value.
`{$mytext := $descrip}`	Appends the string in $descrip.
`{$mytext := $gindex}`	Appends $gindex's numeric value in string form.
`{$mytext := $capturetext}`	Appends text buffer captured by the most recent `{capturetext}` instruction.

Notes: ● Spaces around the := are optional.

● If the appendee variable (to left of :=) is initially numeric (holds a number), the variable will assume the string form, and its updated content will be a string starting with the digits of the number.

● Total max size of a string stored in a variable, after concatenation, is 2000 characters.

Enhancement: Filename now available in pre-/post-processing scripts

Description of problem: At the Publisher main screen, when importing a document instance, you may specify a pre-processing and a post-processing script, along with appropriate arguments to those scripts, which Publisher will action before and after the import. If such a script needs the name of the instance being processed, you must enter that filename as a script argument. Since the filename is already available on the screen, it would be easier and more foolproof for Publisher to unconditionally supply the instance filename to each script invocation

Solution: Publisher now automatically appends the instance name to each pre-process and post-process script call. The name is appended *after* the arguments you supply in the screen box. So, if you enter ‘‘`scriptname field1 field2`’’ in the box, your script will see `field1` as the first argument, `field2` as the second, and the instance name as the third.

Fix: Empty-tag error in valid xml

Description of problem: Processing a validated xml document, Publisher allowed an empty tag to be coded as ⟨tagname/⟩, but disallowed the equally valid form ⟨tagname⟩ ⟨/tagname⟩.

Solution: Publisher now allows either form without error, in a valid xml file.

Fix: **Could not re-import an xml instance.**

Description of problem: Publisher imports a well-formed xml instance, and in doing so it modifies start-tag and end-tag characters to the binary ctrl-O and ctrl-P translations that PubLink's .txt file needs to see, and it does this translation *in the original xml file*. This change to the xml file prevents it being re-imported.

Solution: Publisher's import process has been modified so that the translation of tag-delimiter characters happens only in a copy of the xml file.