Packet Type 65 (0x41)

Graphics Box Style

The “Content” prefix ID referred to in this description, is the standard File Name packet. The “Caption” prefix ID referred to in this description is the standard WP Text packet.

Several of the fields documented in this packet exist only if various bit flags are set. The condition for the existence of a field and a reference to the controlling flag will precede the data. When computing overall sizes and offsets, remember to keep this fact in mind.

If an override bit is set in the box override flags, a mask appears defining which bits within the corresponding data are overridden followed by the actual override data. For example, assume a data byte has two fields, the first field covering bits 0-3 and the second field covering bits 4-7. Assume the first field (bits 0-3) is to be overridden but the second field (bits 4-7) is not. The corresponding override bit will be set in the box override flags word. The data corresponding to this override bit will be a mask byte (value 0x0F) and a data byte, of which only the bottom 4 bits matter.

►

[number of prefix packet IDs] may be 0
- [name/library data PIDs] (none currently defined) see box name/library data PID flag field below
- [counter PIDs (type=0x11)] counter currently defined
- [positioning data PIDs] (none currently defined) see box positioning data PID flag field below
- [content PIDs] (content rendering IDs, actual content not allowed) see box content PID flag field below
- [caption PIDs] WP text packets, (2 possible: default caption initial style, default caption number style; actual caption data not allowed) see box caption PID flag field below
- [border PIDs (type=0x30)] (2 possible: border style and border style overrides) see box border PID flag field below
- [fill PIDs (type=0x44)] (fill style currently defined) see box fill PID flag field below
- [wrap PIDs] (none currently defined) see box wrap PID flag field below
- [hypertext PIDs] (2 possible) see box hypertext PID flag field below

Box name/library data:
[total size of box data] not including this word
[size of box name and library data area] not including this word
<box name/library data PID flags> none currently defined
<box library management flags>
- bit 0: 1 = this is a library style
- bits 1-7: undefined

[box name length word]
- > 0 = user-defined box style; a word string containing the name follows
- 0 = name is "Figure Box" (English only)
- −1 = name is "Table Box" (English only)
- −2 = name is "Text Box" (English only)
- −3 = name is "User Box" (English only)
- −4 = name is "Equation Box" (English only)
- −5 = name is "Button Box" (English only)
- < −5 = undefined box style name
- [word text of name] x ? if box name length > 0

Box counter data:
[size of box counter area]
<box counter PID flags>
bit 0: 1 = counter PID appears at front of style
bits 1-7: currently undefined

Box positioning data:
[size of box positioning area] not including this word
<box positioning PID flags> no bits currently defined
<box general positioning flags>
bits 0-2: box anchor value
0 = page anchor
1 = paragraph anchor
2 = character anchor
bit 3: page offset bit (only used for page anchor boxes)
0 = don't use automatic code placement when inserting
1 = do use automatic code placement when inserting (will delay box code if necessary)
bit 4: overlap flag
0 = box is not overlapping another box
1 = box is overlapping another box
bit 5: auto flag (used in WPwin)
0 = place box to the right of the page
1 = place the box at the cursor
bits 6-7: currently undefined
<box horizontal position flags> horizontal position flags are ignored for character-anchored boxes
bits 0-1: horizontal alignment type
0 = absolute from left of page
1 = aligned to margins
2 = aligned to columns
bits 2-4: horizontal alignment
0 = left
1 = right
2 = center
3 = full
bits 5-7: currently undefined
[box horizontal offset (signed WPU) or absolute position (unsigned WPU)]
<left column> for page anchored boxes aligned to columns
<right column> for page anchored boxes aligned to columns
<box vertical position flags> vertical position flags are ignored for paragraph-anchored boxes
bits 0-1: vertical alignment type (ignored for character-anchored boxes)
0 = absolute from top of page
1 = aligned to margins
bits 2-4: vertical alignment
0 = top
1 = bottom
2 = center
3 = full/baseline (treated as baseline for character boxes)
bit 5: character box vertical effect
0 = box affects line height
1 = box does not affect line height (still affects cursor position)
bits 6-7: currently undefined
[box vertical offset (signed WPU) or absolute position (unsigned WPU)]
<box width flags>
bit 0: width value flag

0 = fixed width (width from width word)
1 = automatic width from content
bits 1-7: currently undefined
[box width (unsigned WPU)]
<box height flags>
bit 0: height value flag
0 = fixed height (height from height word)
1 = automatic height from content
bits 1-7: currently undefined
[box height] unsigned WPU

Box content data:
[size of content area] not including this word
<box content PID flags> not currently defined
<box content type>
0 = no content (empty)
1 = text content (WP7 document)
2 = linked text content (WP7 document)
3 = image (graphics) content (WPG2.0 document)
4 = equation content
5 = presentation content (Draw Presentation document)
6 = real-time video content (not used)
7 = macro content (WPM document)
8 = sound content(WPS document)
128 (0x80) = external content (content rendering/editing is handled by an outside application)
<box content alignment flags>
bits 0-1: horizontal alignment of content
0 = left
1 = right
2 = center
bits 2-3: vertical alignment of content
0 = top
1 = bottom
2 = center
bit 4: if content is scalable, preserve aspect ratio
0 = preserve aspect ratio
1 = don't preserve aspect ratio
bits 5-7: currently undefined
[total size of content rendering data]not including this word
If box content type is empty, there is no rendering data and this word must be 0. This word is the same as the total size word at the front of all the content rendering data definitions.
<box content rendering information> this will be one of the 9 blocks of data defined below under *Box Content Rendering Information* controlled by the box content type field◄

Box Content Rendering Information

The following paragraphs describe the content rendering information for each content type (see the box content type field above). The general form is a word of total length, followed by a word of fixed-size length, followed by the data.

Text Content Rendering Information

Exists if box content type field equals 1.

►
[total size of text rendering information] not including this word
<PID flags>
bit 0: 1 = text content PID appears at front of function. This bit must not be set for a style.
bit 1: 1 = text content default style PID appears at front of style. This bit should not be set in function override data.
bits 2-7: currently undefined
<text content rendering flag>
bit 0: 1 = format text according to simple box rectangle (before corners are applied, and before rotation and translation)
bit 1: currently undefined
bits 2-3: text rotation
0 = text not rotated
1 = rotated 90°
2 = rotated 180°
3 = rotated 270°
bits 4-7: currently undefined◄

Text Content Override

If the text content rendering override bit is set in the box override flags, and the content type of the box is text, the following data is appended to the bottom of the text content rendering flag field:

►
[total size of text content override data] not including this word
[text content override flags] Bits in this flag determine which of the following data exits. If a bit is set, the corresponding data is appended under this flag field. If a bit is 0, the corresponding data is omitted.
bit 15: PID flag override. Corresponding data:
<mask>
<data>
bit 14: Text content rendering flag override. Corresponding data:
<mask>
<data>
bits 13-0: currently undefined◄

Linked Text Content Flags

Exists if box content type field equals 2. Currently not used.

Image Content Rendering Flags

Exists if box content type field equals 3.

►
[total size of graphics rendering information] not including this word
<PID flags>
bit 0: 1 = Image content PID appears at front of function. This bit must not be set for a style.
bits 1-7: currently undefined
[image native width (WPU)]
[image native height (WPU)]
<image content rendering flags>
bits 0-1: area fill handling
0 = normal fills
1 = no area fills are performed (fills are transparent)
2 = non-transparent area fills are always opaque white
bit 2: 1 = Invert colors. The color inversion is performed after the color to

black and white conversion, if black and white bit is set.
bit 3: 1 = colors converted to black and white (black and white threshold)
bits 4-5: available
bit 6: 1 = mirror contents in y
bit 7: 1 = mirror contents in x

{fixed point x scale factor to apply to image} default = 1.0 (scaled to fit in box)
{fixed point y scale factor to apply to image} default = 1.0 (scaled to fit in box)
{signed x translation value to apply to image} (−1.0 to +1.0) interpreted as fraction of untransformed image width. Default = 0, means no translation. Positive translation means image is shifted right.
{signed y translation value to apply to image} (−1.0 to +1.0) interpreted as fraction of untransformed image height. Default = 0, means no translation. Positive translation means image is shifted up.
{fixed point rotation angle to apply to image} (0.0-360.0, counterclockwise) default = 0
<black and white threshold> (0-255) default = 127
The black and white threshold value ranges from 0-255 and simply defines the luminosity cutoff for mapping to black or white. Luminosity is determined by calculating .30 Red + .59 Green + .11 Blue and normalizing to 0-255 range.
{fixed point lightness value} (−1.0 to 1.0) 0 indicates no adjustment
{fixed point contrast value} (−1.0 to 1.0) 0 indicates no adjustment
<image dither method flags>
bits 0-3: dither method
0 = use .PRS specified default
1 = force halftoning
2 = force ordered dither
3 = force error-diffusion dither
bits 4-7: dither source
0 = use .PRS specified default
1 = force WP dithering/halftoning
2 = force printer dithering/halftoning
3 = use external source (such as OLE server under Windows)
{halftone screen LPI} fixed point value, default = −1.0 for LPI is interpreted to mean use the .PRS-specified default.
{plane 0, halftone screen angle} fixed point value, default = −1.0 for the screen angle is interpreted to mean use the .PRS-specified default.
{plane 1, halftone screen angle} fixed point value, default = −1.0
{plane 2, halftone screen angle} fixed point value, default = −1.0
{plane 3, halftone screen angle} fixed point value, default = −1.0
The double word fixed point values are 65536ths. In other words, there is one word of integer and 1 word of fraction (65536ths).
[fractional portion]
[integer portion]◄

Image Content Override

If the image content rendering override bit is set in the box override flags, and the content type of the box is image, the following data is appended after the “plane 3, halftone screen angle” field above:

► [total size of image content override data] not including this word
[image content override flags]
If the labeled bit in the image content override flags field is set, the corresponding data will be appended under this flag field.
bit 15: PID flag override. If this override bit is set, and bit 0 of the PID flags is overridden, the image native size override must also be set.
<mask>
<data>

bit 14: Image native size. This override bit must be set if bit 0 in the image content PID flags is overridden.
[image native width (WPU)]
[image native height (WPU)]
bit 13: image content rendering flags override
<mask>
<data>
bit 12: scaling factor override
{fixed point x scaling factor}
{fixed point y scaling factor}
bit 11: translation factor override
{x translation factor}
{y translation factor}
bit 10: rotation angle override
{data}
bit 9: black and white threshold override
<data>
bit 8: lightness value override
{data}
bit 7: contrast value override
{data}
bit 6: halftone data override
{halftone screen LPI}
{plane 0 halftone screen angle}
{plane 1 halftone screen angle}
{plane 2 halftone screen angle}
{plane 3 halftone screen angle}
bit 5: image dither flags override
<mask>
<data>
bits 4-0: currently undefined◄

Equation Content Rendering Information

Used if box content type field equals 4. This is subject to change.

► [total size of equation rendering data] not including this word
<PID flags>
bit 0: 1 = equation markup text PID appears at front of function. This bit must not be set for a style.
bit 1: 1 = equation compact box format appears at front of function. This bit must not be set for a style. If bit 0 is set (indicating equation markup text PID), bit 2 must be set, even if the PID is invalid (0).
bit 2: 1 = typeface descriptor PID appears at front of style
0 = use current document font for equation. Current document font means the font in effect where the box function appears. For a box template, current document font means the initial document font.
bits 3-7: currently undefined
<equation content rendering flags>
bits 0-7: currently undefined
[equation base font size] in unsigned WPU
<equation color (RGBS)> x 4
{fixed point rotation value to apply to equation} (0.0-360.0, counterclockwise)
Rotation will not be supported in the initial releases of WP7. However, the space for the data will be provided in the equation rendering data for future support.
This data must be initialized to zeros.◄

Equation Content Rendering Data Override

If the content rendering override bit is set in the box override flags, and the content type of the box is equation, the following data is appended to the fixed-point rotation value to apply to equation field above:

► [total size of equation content override data] not including this word
[equation content override flags]
The following data appear according to bits set in the equation content override flags. This corresponding data is appended to this flag field.
bit 15: PID flag override
<mask>
<data>
bit 14: equation rendering flags override
<mask>
<data>
bit 13: equation font size override
[equation base font size]
bit 12: equation color override
<equation color (RGBS)> x 4
bit 11: equation rotation override
{fixed-point rotation angle} (0.0-360.0)
bits 10-0: currently undefined◄

Presentation Content Rendering Information

Used if Box content type field equals 5. This is subject to change.

► [total size of presentation rendering data] not including this word
<PID flags>
bit 0: 1 = Presentation content PID appears at front of function. This bit must not be set for a style.
bits 1-7: currently undefined
<presentation content rendering flags>
bit 0: 1 = run presentation in a full sized window/screen
0 = run presentation inside the box which encloses it◄

Presentation Content Rendering Data Override

Information pending.

Video Content Rendering Information

Used if box content type field equals 6. Currently not used.

► [total size of video rendering data] not including this word
<PID flags>
bit 0: 1 = Video content PID appears at front of function. This bit must not be set for a style.
bits 1-7: currently undefined
<video content rendering flags>
bit 0: 1 = run video in a full sized window/screen
0 = run video inside the box which encloses it◄

Video Content Rendering Data Override

Information pending.

Macro Content Rendering Information
Used if box content type field equals 7. Currently not used.

► [total size of macro rendering data] not including this word
<PID flags>
bit 0: 1 = Macro content PID appears at front of function. This bit must not be set for a style.
bits 1-7: currently undefined
<macro start action>◄

Macro Content Rendering Data Override
Information pending.

Sound Content Rendering Information
Used if box content type field equals 8. Currently not used.

► [total size of sound rendering data] not including this word
<PID flags>
bit 0: 1 = Sound content PID appears at front of function. This bit must not be set for a style.
bits 1-7: currently undefined
<sound start action>◄

Sound Content Rendering Data Override
Information pending.

External Content Rendering Information
Used if box content type field equals 128 (0x80). Currently not used.

► [total size of external rendering data] not including this word
<PID flags>
bit 0: 1 = External content PID appears at front of function. This bit must not be set for a style.
bits 1-7: currently undefined◄

External Content Rendering Data Override
Information pending.

(Graphics Box Style Packet Continued)

► Box caption data:
[size of caption area] not including this word
<box caption PID flags>
bit 0: 1 = Caption data PID appears at front of function. This bit must not be set in a style.
bit 1: 1 = Caption default initial style PID appears at front of style. This should not be set in function override data.
bit 2: 1 = Caption default number style PID appears at front of style. This should not be set in function override data.
bits 3-7: currently undefined
<caption positioning flags>

bits 0-1: side of content that caption appears on.
0 = left
1 = right
2 = top
3 = bottom
bit 2-3: where caption is in relation to border
0 = outside
1 = inside
2 = on
bits 4-5: caption alignment
0 = left or top
1 = right or bottom
2 = center
bit 6: caption alignment offset word
0 = caption alignment offset word is signed WPU
1 = caption alignment offset word is signed 32768ths of box width or height, depending on caption rotation
bit 7: currently undefined

<caption formatting flags>
bits 0-1: formatting width
0 = use caption formatting width word as unsigned WPU
1 = use caption formatting width word as unsigned 65536ths
2 = format caption as if 65536/65536ths, then use maximum line width (automatic size)
bits 2-3: caption rotation
0 = caption not rotated
1 = rotated 90°
2 = rotated 180°
3 = rotated 270°
bits 4-7: currently undefined

[caption formatting width] the units of measure used here depends on the caption formatting flags field above

[caption alignment offset]the units of measure used here depends on the caption formatting flags field above

Box border data:

[size of border area] not including this word

<box border PID flags>
bit 0: 1 = border style PID appears at front of style
bits 1-7: currently undefined

[total size of border template override data] not including this word, minimum = 2
This word is the same as the total size word at the front of the border template override data definitions. It is not an additional word. The following word (fill template override flags) is the same as the word described in the fill template override data definition. It is not an additional flag word.

[border template override flags]

<border template override data> x ? see *Border Template Override Data* below

Box fill data:

[size of fill area] not including this word

<box fill PID flags>
bit 0: 1 = fill style PID appears at front of style
bits 1-7: currently undefined

[total size of fill style override data] not including this word, minimum = 2
This word is the same as the total size word at the front of the fill template override data definitions. It is not an additional word.

[fill style override flags] This word is the same as the word described in the fill template override data definition. It is not an additional flag word.

<fill style override data> x ?

Box wrapping data:
[size of box wrapping area] not including this word
<box wrapping PID flags> no bits currently defined
<box wrapping flags>
bits 0-3: indicates which side text wraps around box
0 = wrap text on side with largest area for text (the most space between side and margin)
1 = wrap text on left side
2 = wrap text on right side
3 = do not wrap text on either side (text flows from above box to below box)
4 = wrap text on both sides with text flow crossing the box for each line
5 = Wrap text on both sides with text flow working in a columnar fashion. This value is not implemented in WP 7. It is treated as wrap across.
6 = no text wrap
bits 4-7: method used to generate the wrapping rectangles.
0 = Generate one wrap rectangle enclosing the entire box. Always relative to the box, never the content.
1 = Generate irregular wrap rectangles via an internal algorithm. If no border style is referenced, generate wrap rectangles from contents; otherwise, generate wrap rectangles from border (including rounded corners) + border outside spacing. If there is a border, the wrap rectangle is relative to the box; otherwise, it is relative to the content.
2 = use user-defined wrap rectangles relative to the box
3 = use user-defined wrap rectangles relative to the content

Box hypertext data:
Hypertext PIDs should not exist in a templates, and the hypertext action type should be none in a template.

[size of hypertext definition area] not including this word
<hypertext PID flags>
bit 0: 1 = filename (doc or macro) PID appears at front of function
bit 1: 1 = bookmark name PID appears at front of function
bits 2-7: currently undefined
<hypertext type flag>
0 = Hypertext action not defined (only value allowed in style)
1 = Hypertext bookmark action
2 = Hypertext macro action◄

Border Template Override Data

If the border template override bit is set in the box override flags, the following data appear:

►
[total size of border template override data] not including this word
[border template override flags]
The following data appear according to bits set in the border template override flags:
bit 15: border template PID flags data
<mask>
<data>
bit 14: border general flags data
<mask>
<data>
bit 13: corner radius data
[corner radius]

bit 12: inside spacing data
[left inside spacing]
[right inside spacing]
[top inside spacing]
[bottom inside spacing]
bit 11: outside spacing data
[left outside spacing]
[right outside spacing]
[top outside spacing]
[bottom outside spacing]
bit 10: drop shadow data
<drop shadow flag>
[drop shadow spacing]
<drop shadow color (RGBS)> x 4
bit 9: border color data
<border color (RGBS)> x 4
bits 0-8: currently undefined◄

Box Fill Override Data

If the box fill override bit is set in the box override flags, the following data appear:

► [total size of fill template override data] not including this word
[border fill override flags]

The following data appear according to bits set in the fill template override flags:
bit 15: box fill PID flags override
<mask>
<data>
bit 14: fill template data override (see *Fill Template Override Data* below)
bits 13-0: currently undefined◄

Fill Template Override Data

If the fill template override bit is set in the box fill override flags, the following data appear:

► [total size of fill template override data] not including this word
[fill template override flags]

The following data appear according to bits set in the fill template override flags:
bit 15: fill colors override
<foreground/start color (RGBS)> x 4
<background/end color (RGBS)> x 4
bits 14-0: currently undefined◄

Box Wrapping Override Data

If the box wrapping override bit is set in the box override flags, the following data appear:

► [total size of box wrapping override data]not including this word
[box wrapping override flags]

The following data appear according to bits set in the box wrapping override flags:

bit 15: box wrapping PID override
<mask>
<data>
bit 14: wrapping flags
<mask>
<data>
bits 13-0: currently undefined◄

Box Hypertext Override Data

If the box hypertext override bit is set in the box override flags, the following data appear:

► [total size of box hypertext override data]not including this word
[box hypertext override flags]

The following data appear according to bits set in the box hypertext override flags:
bit 15: box hypertext PID override
<mask>
<data>
bit 14: box hypertext type flags override
<data>
bits 13-0: currently undefined◄

Format of Platform-Independent Wrap Rectangles

Each wrap rectangle consists of four 16-bit words as follows:

► [unsigned offset of left side of rectangle from left side of formatted render box]
[unsigned offset of right side of rectangle from left side of formatted render box]
[unsigned offset of top side of rectangle from top side of formatted render box]
[unsigned offset of bottom side of rectangle from top side of formatted render box]◄

Units for the above values are in 65536ths of the appropriate dimension (width or height) of the formatted box in which the content is to be rendered. The left offset value and the right offset value are measured in 65536ths of the width of the formatted box. The top offset value and bottom offset value are measured in 65536ths of the height of the formatted box. The right value must be greater than or equal to the left value and the bottom value must be greater than or equal to the top value. Allowable values are from 0 to 65535. For convenience, 65535 is considered 65536.

General Rules

All currently undefined data should be set to 0 when a new box template or box function is created.

For any flag data, only those bits currently defined should be modified. All others should be preserved.

Code should be written to preserve fixed and variable length data that may be added later. This means utilizing the value of the total length and the values of the lengths of individual data sections.

Names of templates will not be longer than 80 characters (160 bytes), not including length word.

Packet Type 66 (0x42)

Border Line Style

The following format shows all possible fields. Some of the fields may or may not exist depending on various flag bits.

► [number of PIDs (minimum 0)]
    [name/library data PID] none currently defined
    [line data PIDs] none currently defined
[total size of line style data] not including this word nor any previous PID data

Line name and library data area:
[size of line name and library data] not including this word
<line name/library data PID flags> none currently defined, controls name/library data PID above
<line library management flags>
    bit 0: 1 = this is a library style
    bits 1-7: undefined
[line name length]
    > 0 = length of style name in bytes
    0 = single line (English only)
    −1 = double line (English only)
    −2 = dashed line (English only)
    −3 = dotted line (English only)
    −4 = thick line (English only)
    −5 = extra thick line (English only)
    −6 = thin/thick double line (English only)
    −7 = thick/thin double line (English only)
    −8 = button top/left line (English only)
    −9 = button bottom/right line (English only)
    -10 = heavy line (English only)
    -11 = heavy double line (English only)
[line name] x ? exists only if line name length > 0◄

Line data area:

► [size of line data] not including this word
<line data PID flags> none currently defined, controls line data PIDs above
[total width of line definition (WPUs)]
<number of parallel line segments defined> minimum = 1
    When the line definition is used in a border, the first line defined is the innermost and the last line defined the outer-most. When the line definition is used for a horizontal line, the first line defined is the top-most and the last line defined is the bottom-most. When the line definition is used for a vertical line, the first line defined is the left-most and the last line defined is the right-most.
The following is repeated for each line piece defined:
[size of line segment definition] not including this word
<line segment general flags>
    bits 0-1: style type
        0 = predefined WPG line style (see *Predefined WPG Line Style Data* below)
        1 = custom (user-defined) line style, data in segment definition

bits 2-7: currently undefined
[width (thickness) of line (WPU)] may be zero
[spacing between this border line and the next] Spacing is measured between the two printable edges of the lines next to each other.
<line segment (RGBS)> x 4
<line segment style definition> x ? data depends on value in general flags (see *Line Segment Styles* below)◄

Line Segment Styles

One of the following data fields appears in the style definition field above, based on the style type field in the line segment general flags.

Predefined WPG line style data (Exists if the line segment general flags field equals 0):

► <WPG line style set> 0 for now
<WPG line style Number>◄

Custom line style data (Exists if the line segment general flags field equals 1):

► <number of on/off commands> (minimum = 2) If the count is zero or 1, the line is solid. Therefore the minimum must be 2. The commands alternate between on and off run lengths for drawing the line.
[on/off commands] (run lengths in unsigned WPUs, see *Rendition Options* in the *WordPerfect Graphics 2.0 File Format Developer Reference* for examples)◄

Packet Type 67 (0x43)

Fill Style

The following format shows all possible fields. Some of the fields may or may not exist depending on various bit flags.

► [number of child PIDs] minimum = 0
[name/library data PIDs] none currently defined
[fill data PIDs] none currently defined
[total size of fill style data] not including this word
[size of fill name and library data area] not including this word]
<fill name/library data PID flags> none currently defined
<fill library management flags>
bit 0: 1 = this is a library style
bits 1-7: undefined
[fill name length word]
> 0 = length of fill style name text in bytes
0 = 10% fill (English only)
−1 = 20% fill (English only)
−2 = 30% fill (English only)
−3 = 40% fill (English only)
−4 = 50% fill (English only)
−5 = 60% fill (English only)
−6 = 70% fill (English only)
−7 = 80% fill (English only)
−8 = 90% fill (English only)
−9 = 100% fill (English only)
−10 = button fill (English only)

<word text of name> x ? present only if fill name length > 0
[size of fill style first variable-length area] not including this word
<fill PID flags> none currently defined
<color 1 (RGBS)> x 4 foreground for WPG2, starting for gradient
<color 2 (RGBS)> x 4 background for WPG2, ending for gradient
<fill type>
0 = WPG2 pattern set/index
1 = gradient fill
[size of fill information] not including this word
<variable fill information> x ? depends on type (see *Fill Types Data* below)

Fill Types Data

Depending on how the value contained in the fill type field, one of the two following data blocks is inserted into the fill information field above.

WPG2 Pattern Fill

Used if fill type field equals 0.

☺⌐ <WPG2 pattern set> always 0
<WPG2 pattern index>◄

Gradient Fill

Used if fill type field equals 1.

► <flags>
bits 0-4: Gradient type. These correspond exactly to the gradient types the graphical interface supports.
0 = linear
1 = radial
2 = rectangular
bit 5: 1 = rectangular or radial gradient has same aspect ratio as document object
0 = square or circular
bits 6-7: currently undefined
[number of gradient steps] 0 means calculate number of steps
{fixed point gradient rotation angle} (0.0-360.0)
[offset] fractional (65536ths) offset of gradient starting point from left of object bounding box (for convenience, 65535 is considered 65536)
[fractional (65536ths) y offset of gradient starting point from top of object bounding box](for convenience, 65535 is considered 65536)◄

Packet Type 68 (0x44)

Border Style

The following format shows all possible fields. Some of the fields may or may not exist, depending on various bit flags.

► [number of PIDs] minimum = 1
[left side line style PID (type=0x42)]
[right side line style PID (type=0x42)]
[top side line style PID (type=0x42)]
[bottom side line style PID (type=0x42)]
[separator side line style PID (type=0x42)]
[total size of border style data] not including this word
[size of border name and library data area] not including this word
<border name and library data PID flags> none currently defined
<border library management flags>
bit 0: 1 = this is a library style

[border name length word]
> 0 = length of border style name text in bytes
0 = spacing only (no lines)
−1 = single border (English only)
−2 = double border (English only)
−3 = dashed border line (English only)
−4 = dotted border (English only)
−5 = thick border (English only)
−6 = extra thick border (English only)
−7 = thin/thick double border (English only)
−8 = thick/thin double border (English only)
−9 = thick top and bottom border (English only)
−10 = button border (English only)
<word text of name> x ? present if border name length word > 0
[size of border data area] not including this word
<border PID flags>
bit 0: 1 = left side line PID appears at front of style
bit 1: 1 = right side line PID appears at front of style
bit 2: 1 = top side line PID appears at front of style
bit 3: 1 = bottom side line PID appears at front of style
bit 4: 1 = separator line PID appears at front of style
bits 5-7: currently undefined
<general definition flags>
bit 0:
0 = use colors from individual segment definitions
1 = user border color for line colors
bit 1:
0 = use autospacing for inside and outside spacing
1 = use inside and outside spacing fields which follow
bits 2-7: currently undefined
[corner radius (WPUs)]
[left inside spacing]
[right inside spacing]
[top inside spacing]
[bottom inside spacing]
[left outside spacing]
[right outside spacing]
[top outside spacing]
[bottom outside spacing]
[separator outside spacing] Separator spacing is added to both sides of the separator line.
<border color (RGBS)> x 4
<drop shadow flag>
0 = no drop shadow effect
1 = drop shadow toward upper left corner
2 = drop shadow toward lower left corner
3 = drop shadow toward lower right corner
4 = drop shadow toward upper right corner
[drop shadow spacing (WPU)]
<drop shadow color (RGBS)> x 4◄

Packet Type 69 (0x45)

Graphics Rule Counter

This packet has the same format as the Counters Data packet 17 (0x11).

►
[name] x ? null terminated
<number of counter levels valid (1-5)>

<default numbering method for each level> x 5
[level name 1] x ? null terminated
[level name 2] x ? null terminated
[level name 3] x ? null terminated
[level name 4] x ? null terminated
[level name 5] x ? null terminated◄

Packet Type 70 (0x46)
Intellitag TAG with Attributes

►
[number of attributes]
[attribute 1 prefix ID] prefix ID's of attribute values (child packets)
[attribute 2 prefix ID]
...
[attribute n prefix ID]
[tag error flag] error flag that encompasses full tag (0 = no error)
- bit 0: tag length is exceeded
- bit 1: attribute specification length exceeded
- bit 2: number of IDs in attribute values exceeded
- bit 3: number of entities in attribute values exceeded

The following information is repeated for each attribute:
- [attribute DTD index] can get type of attribute from index number
- <attribute error code>
  - bit 7: 1 = both LITDEL and ALITDEL are in the attribute value
  - bits 0-6:
    - 0 no error
    - 1 attribute is unknown to DTD
    - 2 illegal character in attribute name value
    - 3 illegal character in attribute name token value
    - 4 illegal character in attribute number value
    - 5 illegal character in attribute number token value
    - 6 value in attribute exceeds NAMELEN limit
    - 7 too many tokens in attribute
    - 8 value in attribute exceeds LITLEN limit
    - 9 no mapping for extended character in attribute
    - 10 attribute has empty attribute value literal
    - 11 token value not in declared value list
    - 12 mismatch between DTD fixed value and attribute value
    - 13 unknown entity type attribute value
    - 14 illegal entity type attribute value
    - 15 entity open in attribute value not expanded
    - 16 character reference open in attribute value not checked
    - 17 attribute is duplicated in tag◄

Packet Type 71 (0x47)
IntelliTag® Attribute Text

►
[number of blocks = 1]
{offset to text}
{size of text}
<lit delimiter> 1 = LITDEL present, 2 = ALITDEL present, 3 = both present
[attribute name] x ?
<text> x ?◄

Packet Type 72 (0x48)
IntelliTag LGC Filename

► [number of child PIDs = 3]
[child ID for LSI filename]
[child ID for Document Type name]
[child ID for Alias filename]
<Reserved> x 6
<LGC filename> x ?◄

Packet Type 73 (0x49)
IntelliTag External File Reference

► [number of child PIDs = 2]
[notation attribute PID]
[system identifier (filename) PID]
[data type] CDATA, SDATA, and so forth
[entity declaration error] error flag for entire declaration
[number of text strings] currently 3
[size of entity name]
[size of public identifier]
[size of notation name]
[external entity name] x ?
[public identifier] x ?
[notation name] x ?◄

Packet Type 74 (0x4A)
IntelliTag Notation Attributes

► [number of attributes]
[attribute 1 PID] prefix IDs of attribute values
[attribute 2 prefix ID]
...
[attribute n prefix ID]
[notation error flag] error flag for this set of attributes (see packet 0x46)
The following information is repeated for each attribute:
[attribute 1 DTD index] can get type of attribute from index number
<attribute 1 error code> error for each attribute (see packet 0x46)◄

☺ ♦♣ Packet Type 75 (0x4B) ☻♦
IntelliTag DTD Entity Reference

► [entity name] x ?◄

Packet Type 76 (0x4C) - 82 (0x52)
Reserved

Packet Type 83 (0x53)
Windows Data Store Directory
No documentation will be provided for this packet.

Packet Type 84 (0x54)
Windows Data Store Data
No documentation will be provided for this packet.

Packet Type 85 (0x55)
Desired Font Descriptor

For further description of the primary family ID, scripting system, width, weight, attributes, and so forth, see the documentation for packet type 32 (0x20), which has the same format as this packet.

►
[average character width (WPU)]
[short ascender height (PSU)]
[x height (PSU)]
[descender height (PSU)]
[italic adjust (PSU)]
[primary family ID]
<scripting system>
<primary character set>
<width (aspect ratio)>
<weight>
<attributes>
<general characteristics>
<classification>
<fill byte>
<font type>
<font source file type>
[typeface name length] maximum 58
<typeface name> x ? WP word string◄

When no printer is selected, the default font descriptor is the following:

►
[0x3C = average character width]
<0x1e14 = ascender height (77%)>
<0x170C = x height>
<0x0A8C = descender height>
<0 = italic adj>
<1 = Courier family ID>
<1 = scripting system>
<0 = primary character set (ASCII)>
<0x70 = width (aspect ratio)>
<0x60 = weight>
<0 = attributes (normal)>
<0x25 = general characteristics>
<0 = serif classification>
<0 = built-in font type>
<0x10 = .PRS source>
<0 = typeface name length for default descriptor>
<typeface name = does not exist for default descriptor>◄

Packet Type 86 (0x56)

Merge File Type

►
[merge file type]
1 = primary document
2 = table secondary type merge document
4 = secondary merge type document◄

Packet Type 87 (0x57)

Sound Clip

►
[number of child prefix IDs = 1]

[child file packet PID (type=0x58)]
<sound clip type>
0 = MIDI sound
1 = digital audio
others to be defined
<left volume> in units of dB
<right volume> in units of dB
[sound clip description] x ? null terminated◄

Example 1:

Packet type 0x57 points to a child packet type 0x58 (Sound Clip Child Packet). This child packet 0x58 contains a filename and may point to one or more child packets which are also packet type 0x58. These child packets contain sound data.

Example 2:

Packet type 0x57 points to a child packet type 0x07 (Native Filename). This child packet 0x07 contains a filename and may point to one or more child packets which are packet type 0x6F (Graphics Data). These child packets contain sound data in this case.

Both of these examples are supported by WordPerfect.

Packet Type 88 (0x58)

Sound Clip Child Packet

Contains a filename or a data packet. If this packet does not have any children, the format is:

►
[sound data filename] x ? null terminated
or
<sound data> x ?◄

If this packet does have children, the format is the following:

►
[number of child IDs]
[ID 1 (type=0x58)] ID of child packet containing sound data
[ID 2 (type=0x58)] ID of child packet containing sound data
...
[ID n (type=0x58)] last child ID containing sound data
[tag 1] tag flag for first child packet
[tag 2] tag flag for second child packet
...
[tag n] last tag for last child ID
[sound filename] x ?null terminated◄

Format for Tag Flags

0x0001 = data being kept internal
0x0010 = data being kept internal (hot link)
0x0011 = data being kept internal (warm link)

0x0012 = data being kept internal (cold link)

Packet Type 89 (0x59)

Merge Associated Data File

► [merge associated filename] x ?◄

Packet Type 90 (0x5A)

Reserved

Packet Type 91 (0x5B) -95 (0x5F)

Available

Packet Type 96 (0x60)

Text/Single Cell Name

Holds floating cell name.

►
[length of cell name packet]
[count] always = 0 for single cell packets
[cell name word string] x 21(max) null terminated◄

Packet Type 97 (0x61)

Table Names

This packet holds one or more table names.

►
[length of table names packet]
[count of names which follow the table name]
The count of names will be 0 if no data follows the table name. If count of names is greater than zero, the structure data following the table exists. The data following the table name is a repeating structure for names of cells, columns, rows, and blocks.
[table name] x 21(max) null terminated
[length of this name structure]
[reference name] x ?null terminated
<reference type>
1 = row
2 = column
3 = cell
4 = block◄

The data which follows depends on the reference type.

►
Row Type (1):
[row number]
Column Type (2):
[column number]
Cell Type (3):
[row number of Cell]
[col number of Cell]
Block Type (4):
[begin row number of block]
[begin col number of block]

[end row number of block]
[end col number of block]◄

Packet Type 98 (0x62)

Page Number Format String

► [page number format string] x ?◄

Packet Type 99 (0x63)

Math Formulas for Columns

► <constants and operators> x ?◄

Math Operators

Data are stored in postfix or Reverse Polish Notation (RPN) order. All postfix math rules apply. Operators are stored in the following WordPerfect-specific codes:

Operator		**Code**
Add	(+)	1 (0x1)
Subtract	(−)	2 (0x2)
Multiply	(*)	3 (0x3)
Divide	(/)	4 (0x4)
Negative	(−)	5 (0x5)
Positive	(+)	6 (0x6)
Average	(+/)	7 (0x7)
Equal	(=)	8 (0x8)
Average Totals	(=/)	9 (0x9)

Constants

Each (decimal) digit of a constant is stored as a hexadecimal value in a nibble (4 bits, 2 nibbles per byte). The first nibble of the constant equals 12 (0xC) and marks the beginning of the number. Following this nibble, each digit is stored as that digit plus 1 [that is, 0 = 1 (0x1), 1 = 2 (0x2), 9 = 10 (0xA)] and occupies one nibble. If the number occupies fewer than seven digits, a 15 (0xF) follows the last digit. A seven-digit number has no 15 (0xF) value on the end. The 14 (0xE) value represents a decimal point. WordPerfect puts 1 (0x1) into unused bytes that follow 15 (0xF), but these unused bytes are not required to read the value of the constant. Examples are shown in the table below.

Hex Bytes	**Decimal Value**
C2 3E 4F 01	12.3

Hex Bytes	Decimal Value
C1 F0 01 01	0
C3 51 2F 00	2,401 (in the last unused byte [2 nibbles], 00 is equivalent to 01)
CA 11 11 11	9,000,000
CE 42 71 F0	.3160
C1 E4 27 1F	0.3160
CA 9E 87 F0	98.76

Packet Type 100 (0x64)

Equation Compact Box

► <node type>
<node contents> x ?
<node type>
<node contents> x ?
...
...◄

Node types:

► 1 = attribute type (size = 8)
[scale x]
[scale y]
[attributes]
[font index]
0 = base font
1 = super/sub font
2 = double super/sub font

2 = box type (size of fixed portion of box type = 5)
[x position (relative to equation box)]
[y position (relative to equation box)]
<length of string>
<string> x ?

3 = font type (three font descriptors, each 139 bytes long)
<font descriptor for base font> x 139
<font descriptor for subscript/superscript font> x 139
<font descriptor for double subscript/superscript font> x 139

4 = diacritical type (same as box but for diacritical chars)
[x position (relative to equation box)]
[y position (relative to equation box)]
<length of string>
<string> x ?

5 = line type (line type box for underline) (size = 8)
[x position (relative to equation box)]
[y position (relative to equation box)]
[width of box]

[height of box]◄

Packet Type 101 (0x65)

Reserved

Format before printing flag. Used by WP formatter. Invalid in disk files.

Packet Type 102 (0x66)

Unique Table ID

The packet is repeated for each table that exists in a document. It contains only one word.

► [unique table ID] table creation count◄

Packet Type 103 (0x67)

Unique Floating Cell ID

The packet is repeated for each floating cell that exists within a document. It contains only one word.

► [unique floating cell ID] floating cell creation count◄

Packet Type 104 (0x68)

Table Database

This packet is used by WordPerfect for Windows and is not used in DOS. No documentation will be provided for this packet.

Packet Type 105 (0x69)

Table Style

The data in this packet will appear in the order shown below. Much of the data is optional and exists only if a corresponding flag bit is set.

► [number of PIDs]
[table border ID] check bit 0 of [PID flags]
[table default lines IDs] check bit 1 of [PID flags]
[table border override line IDs] present for each [table border override line IDs flags] bit set
[default line override IDs] present for each [table default line IDs flags] bit set
[last row line IDs] present for each [last row border IDs flags] bit set
[first column line IDs] present for each [first column border IDs flags] bit set
[last column line IDs] present for each [last column border IDs flag] bit set
[header rows line IDs] check bit 0 of [header row flags] for each header row
[table fill ID] check bit 2 of [PID flags]
[table alternate fill ID] check bit 3 of [PID flags]
[last row fill ID] check bit 7 of [PID flags]
[first column fill ID] check bit 9 of [PID flags]
[last column fill ID] check bit 11 of [PID flags]
[header rows fill IDs] check bit 1 of [header row flags] for each header row
[size of all of the following data] excluding this word
[size of name information]
<name/library PID flags> none currently defined.
<library management flags>

[length of table style name] negative number indicates predefined table style
<table style name> x ? exists if length > 0
[size of rest of data] excluding this word
[PID flags] border and fill PIDs exist if bit is set
bit 0: 1 = table border ID exists
bit 1: 1 = table default lines ID exists
bit 2: 1 = table fill ID exists
bit 3: 1 = table alternate fill ID exists
bit 4: 1 = table border override lines IDs exist
bit 5: 1 = default line override IDs exist
bit 6: 1 = last row line IDs exist
bit 7: 1 = last row fill ID exists
bit 8: 1 = first column line IDs exist
bit 9: 1 = first column fill ID exists
bit 10: 1 = last column line IDs exist
bit 11: 1 = last column fill ID exists
bit 12: 1 = drop shadow exists
[override flags] border, fill, and format overrides exist if bit is set
bit 0: 1 = table border overrides exist
bit 1: 1 = default line overrides exist
bit 2: 1 = table fill overrides exist
bit 3: 1 = table alt. fill overrides exist
bit 4: 1 = last row line overrides exist
bit 5: 1 = last row fill overrides exist
bit 6: 1 = first column line overrides exist
bit 7: 1 = first column fill overrides exist
bit 8: 1 = last column line overrides exist
bit 9: 1 = last column fill overrides exist
bit 10: 1 = table format overrides exist
bit 11: 1 = last row format overrides exist
bit 12: 1 = first column format overrides exist
bit 13: 1 = last column format overrides exist
[number of header rows]
[starting header row number] zero based
<alternating fill flag>
0 = no alternate fill
1 = rows
2 = columns
3 = both rows and columns
[number of times to do primary fill]
[number of times to do alternating fill]◄

The remaining data is present only if the associated flag bits are set.

► [table border override line IDs flags] check bit 4 of [PID flags]
bit 0: 1 = left line PID exists
bit 1: 1 = right line PID exists
bit 2: 1 = top line PID exists
bit 3: 1 = bottom line PID exists
bit 4: 1 = inside/separator line PID exists
<table border overrides> x 6 check bit 0 of [override flags] (see *Border Override Data* below)
[table default line IDs flags] check bit 5 of [PID flags]
bit 0: 1 = left line PID exists
bit 1: 1 = right line PID exists
bit 2: 1 = top line PID exists
bit 3: 1 = bottom line PID exists
bit 4: 1 = inside/separator line PID exists

<table default line overrides> x 6 check bit 1 of [override flags] (see *Border Override Data* below)
<table fill overrides> x 10 check bit 2 of [override flags] (see *Fill Override Data* below)
<table alternate fill overrides> x 10 check bit 3 of [override flags] (see *Fill Override Data* below)
<table format data> x 11 check bit 10 of [override flags] (see *Format Data* below)
<drop shadow overrides> x 7 check bit 12 of [PID flags] (see *Drop Shadow Data* below)
[last row border IDs flags] check bit 6 of [PID flags]
bit 0: 1 = left line PID exists
bit 1: 1 = right line PID exists
bit 2: 1 = top line PID exists
bit 3: 1 = bottom line PID exists
bit 4: 1 = inside/separator line PID exists
<last row border overrides> x 6 check bit 4 of [override flags] (see *Border Override Data* below)
<last row fill overrides> x 10 check bit 5 of [override flags] (see *Fill Override Data* below)
<last row format data> x 11 check bit 11 of [override flags] (see *Format Data* below)
[first column border IDs flags] check bit 8 of [PID flags]
bit 0: 1 = left line PID exists
bit 1: 1 = right line PID exists
bit 2: 1 = top line PID exists
bit 3: 1 = bottom line PID exists
bit 4: 1 = inside/separator line PID exists
<first column border overrides> x 6 check bit 6 of [override flags] (see *Border Override Data* below)
<first column fill overrides> x 10 check bit 7 of [override flags] (see *Fill Override Data* below)
<first column format data> x 11 check bit 12 of [override flags] (see *Format Data* below)
[last column border IDs flags] check bit 10 of [PID flags]
bit 0: 1 = left line PID exists
bit 1: 1 = right line PID exists
bit 2: 1 = top line PID exists
bit 3: 1 = bottom line PID exists
bit 4: 1 = inside/separator line PID exists
<last column border overrides> x 6 check bit 8 of [override flags] (see *Border Override Data* below)
<last column fill overrides> x 10 check bit 9 of [override flags] (see *Fill Override Data* below)
<last column format data> x 11 check bit 13 of [override flags] (see *Format Data* below)◄

The following flags word is included for each header row in the table. The flags indicate what data exists after the flags word.

► [header row flags]
bit 0: 1 = border line IDs exist
bit 1: 1 = fill pattern ID exists
bit 2: 1 = border overrides exist
bit 3: 1 = fill overrides exist
bit 4: 1 = format data exists
[line IDs flags] check bit 0 of [header row flags]
[fill pattern ID exists] check bit 1 of [header row flags]
<border overrides> x 6 check bit 2 of [header row flags] (see *Border Override Data* below)
<fill overrides> x 10 check bit 3 of [header row flags] (see *Fill Override Data* below)
<format data> x 11 check bit 4 of [header row flags] (see *Format Data* below)◄

Border Override Data

► [use flags]
- bit 0: red
- bit 1: green
- bit 2: blue
- bit 3: shade

<border line override color (RGBS)> x 4◄

Fill Override Data

► [use flags]
- bit 0: foreground
- bit 1: background

<fill override foreground color (RGBS)> x 4

<fill override background color (RGBS)> x 4◄

Format Data

► [use flags]
- bit 0: attributes
- bit 1: number type
- bit 2: currency index
- bit 3: justification
- bit 4: join cells in row or column

[attribute word 1]
- bit 0: 1 = extra large
- bit 1: 1 = very large
- bit 2: 1 = large
- bit 3: 1 = small print
- bit 4: 1 = fine print
- bit 5: 1 = superscript
- bit 6: 1 = subscript
- bit 7: 1 = outline
- bit 8: 1 = italics
- bit 9: 1 = shadow
- bit 10: 1 = redline
- bit 11: 1 = double underline
- bit 12: 1 = bold
- bit 13: 1 = strikeout
- bit 14: 1 = underline
- bit 15: 1 = small caps

[attribute word 2]
- bit 0: 1 = blink
- bit 1: 1 = reverse video

[number type]
- bits 0-3: number of digits to display after decimal point or date index information
- bit 4: 1 = display with commas
- bit 5: rounding
  - 0 = use full precision in calculations
  - 1 = use displayed precision
- bits 6-7: negative numbers
  - 0 = '-' (use minus sign)
  - 1 = ( ) (use parentheses)
  - 2 = CR/DR (use credit/debit symbols)
  - 3 = not defined
- bits 8-11: standard formats
  - 0 = general
  - 1 = integer

2 = fixed
3 = percent
4 = currency
5 = accounting
6 = commas
7 = scientific
8 = date
9 = text

bit 12: 1 = text
bit 13: 1 = set currency symbol on
bits 14-15: notation
0 = floating point
1 = scientific
2 = fixed
3 = not defined

<currency symbol index>
bits 0-6: index of the desired currency, default = 0
bit 7: alignment

[justification]
0 = left
1 = full
2 = center
3 = right
4 = all (kinto waritsuke)
5 = decimal align◄

Drop Shadow Data

► <drop shadow flag>
0 = no drop shadow
1 = upper left corner
2 = lower left corner
3 = lower right corner
4 = upper right corner

[drop shadow spacing]

<drop shadow color (RGBS)> x 4◄

Packet Type 106 (0x6A) - 109 (0x6C)
Reserved

Packet Type 109 (0x6D)
World Wide Web

► [number of PIDs = 2]
[base URL PID (type=0x07)]
[background image URL PID (type=0x07)]

[flags]
bit 0: 1 = document is an Internet Publisher Document
bit 1:
0 = use first heading in document for HTML title
1 = use Document Summary Descriptive Name for HTML title

<text color (RGB)>
<hypertext color (RGB)>
<visited hypertext color (RGB)>
<active hypertext color (RGB)>
<background color (RGB)>◄

Packet Type 110 (0x6E)

Conversion Font Descriptor

5.*x* and 7 fonts which are used only for conversion purposes are stored. The equivalent 5.*x* packet is 0x30. When a document is converted from 7 to 5.*x*, the native 7 fonts are stored in 0x30 packets. When a document is converted from 5.*x* to 7, then the native 5.x fonts are stored in packet type 0x6E. This packet type was first used in 6.0a for Windows and 6.0b for DOS.

Packet Type 111 (0x6F)

Graphics Data

Contains WPG graphics data. See the *WordPerfect Graphics 2.0 File Format Developer Reference* for details of the graphic format.

Packet Type 112 (0x70)

OLE Object Descriptor

No documentation will be provided for this packet.

Packet Type 113 (0x71)

OLE Object Data

No documentation will be provided for this packet.

Packet Type 114 (0x72)

File Theta

Theta associated with this file. No documentation will be provided for this packet.

Packet Type 115 (0x73)

DDE Link Information

This packet will have at least one child packet defined. Any child packets will have a tag flag defining the platform of the DDE Data packet. All windows platforms use the Windows platform tag (0x9800).

► [count (n) of child PIDs]
[child PID] x n PID for each child packet
[tag flags] x n platform private tag for each child packet
0x8000 = DG
0x8400 = Macintosh
0x8800 = NeXT
0x8C00 = PC (MS-DOS)
0x9000 = UNIX
0x9400 = VAX
0x9800 = Windows
0x9C00 = IBM 370
0xA000 = OLE
[filename] x ? null-terminated word string defining the user name of the DDE link◄

Packet Type 116 (0x74)

DDE Link Data

► <link storage and update options>
bit 0: 1 = text (bits 0 and 1 are mutually exclusive)
bit 1: 1 = graphics
bit 2: must be 0
bit 3: must be 0
bit 4: 1 = manual (bits 4 and 5 are mutually exclusive)
bit 5: 1 = automatic
bit 6: must be 0
bit 7: must be 0
[default format for link]
{date time stamp (seconds)} reserved, set to NULL.
[date time stamp (year)] reserved, set to NULL.
[link name] x 128 This null-terminated string should be the same as the filename in packet type 116 (0x73).
<link source> x 384 This null-terminated string should contain the server application, topic and item names. The names are to be separated by a vertical bar (|) such as server|topic|item.
<link source document name> x 64 null-terminated string◄

The [default format for link] (DFL) and <link source document name> (LSDN) items are mutually exclusive. If the format is predefined, the DFL should be the predefined constant. If the format is not defined, the LSDN should have the name that would be returned from the Windows API function GetClipboardFormatName( ). If the format is unknown, then the DFL should be 0 and the LSDN a null string. WordPerfect will then negotiate the preferred format type when the document is loaded the first time.

Packet Type 117 (0x75)

Template Description

No documentation will be provided for this packet.

Packet Type 118 (0x76)

Macintosh-Specific Data

No documentation will be provided for this packet.

Packet Type 119 (0x77)

Formatter Undo State

► [undo levels] number of undos in document
[redo levels] number of redos in document (The structures for redos are the same as the undos, but will start after the last undo index.)◄

The next 6 bytes are repeated for each undo and redo level.

► [undo level] level number at this index
[undo count] number of undos at this level
[undo token] token that triggered this undo (margin change, text, and so forth)◄

Packet Type 120 (0x78)

Reserved

Packet Type 121 (0x79)

Reserved

Packet Type 122 (0x7A)

Reserved

Packet Type 123 (0x7B)

Swapped Style Undo

This packet is the same format as Normal Style packet type 0x30. It is used to undo a style change.

Packet Type 124 (0x7C)

OBEX ID

ID string for export of OBEX data.

► {0xFFFFFFFF} region ID for the whole document
<Obex object ID> x ? null-terminated ANSI string format◄

Packet Type 125 (0x7D)

SGML Tag with Attributes (Parent Packet)

► [number of attributes]
[attribute 1 PID] PIDs of attribute values (child packets)
[attribute 2 PID]
...
[attribute n PID]
[tag error flag] bitwise error flag for the entire tag
0 = no error
1 = tag length is exceeded for the tag
2 = attribute specification length exceeded for the tag
4 = number of IDs in attribute values is exceeded
8 = number of entities in attribute values is exceeded◄

The following information is repeated for each individual attribute.

► [attribute 1 DTD index] index to attribute information in memory
<attribute 1 error code> error for individual attribute
Only bit 7 is bitwise. If bit 7 is set then both LITDEL and ALITDEL is in the attribute value.
0 = no error
1 = attribute is unknown to DTD
2 = illegal character in attribute name value
3 = illegal character in attribute name token value
4 = illegal character in attribute number value
5 = illegal character in attribute number token value
6 = value in attribute exceeds name length limit
7 = too many tokens in attribute
8 = value in attribute exceeds LITLEN limit
9 = no mapping for extended character in attribute
10 = attribute has empty attribute value literal
11 = token value not in declared value list

12 = mismatch between DTD fixed value and attribute value
13 = unknown entity type attribute value
14 = illegal entity type attribute value
15 = entity open in attribute value not expanded
16 = character reference open in attribute value not checked
17 = attribute is duplicated in tag
[attribute 2 DTD index]
<attribute 2 error code>
...
[attribute n DTD index]
<attribute n error code>◄

Packet Type 126 (0x7E)

SGML Attribute Value (Child Packet)

► [number of blocks = 1]
{offset to text}
{size of text}
<literal delimiter> 1 = LITDEL, 2 = ALITDEL, 3 = both present
[attribute name] x ?
<text> x ?◄

Packet Type 127 (0x7F)

SGML LGC Filename

If this packet ever has child IDs, the format will be that of the Native Filename packet type 0x07.

► [LGC filename] x ? null terminated◄

Packet Type 128 (0x80)

SGML External Entity

► [number of child IDs = 1]
[ID 1 (type=0x81)] ID of child packet for notation attributes
[data type] CDATA, SDATA, and so forth
[entity declaration error] error flag for full declaration
[number of text strings]
[size of entity name]
[size of file path (system ID)]
[size of public ID]
[size of notation name]

[external entity name] x ?
[file path (system ID)] x ?
[public ID] x ?
[notation name] x ?◄

Packet Type 129 (0x81)

SGML Notation with Attributes

Identical to SGML Tag with Attributes packet (0x7D).

► [number of attributes]
[attribute 1 PID] PIDs of attribute values
[attribute 2 PID]

...
[attribute n PID]
[notation error flag] error flag for entire notation (see packet type 0x7D)
The following information is repeated for each individual attribute:
[attribute DTD index] index to attribute information in memory
<attribute error code> error for individual attributes (see packet type 0x7D)◄