Paint Shop Pro (PSP)
File Format Specification

September, 2005

Paint Shop Pro (PSP) File Format Version 8.0
Paint Shop Pro Application Version 10.0
File Format Documentation Version 8.0

By downloading and / or using these Specifications, user certifies that they have read and agreed to the following disclaimers and warranties:

The Paint Shop Pro File Format Specification (the "Specification") is copyrighted 1998 to 2005 by Corel Corporation ("Corel"). Corel grants you a nonexclusive license to use the Specification for the sole purpose of developing software products(s) incorporating the Specification. You are also granted the right to identify your software product(s) as incorporating the Paint Shop Pro Format, provided that your software in incorporating the Specification complies with the terms, definitions, constraints and specifications contained in the Specification and is subject to the following:

DISCLAIMER OF WARRANTIES. THE SPECIFICATION IS PROVIDED "AS IS." COREL DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.

You are solely responsible for the selection, use, efficiency and suitability of the Specification for your software products.

OTHER WARRANTIES EXCLUDED. COREL SHALL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, EXEMPLARY, PUNITIVE OR INCIDENTAL DAMAGES ARISING FROM ANY CAUSE, EVEN IF COREL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. CERTAIN JURISDICTIONS DO NOT PERMIT THE LIMITATION OR EXCLUSION OF INCIDENTAL DAMAGES, SO THIS LIMITATION MAY NOT APPLY TO YOU.

IN NO EVENT WILL COREL BE LIABLE FOR ANY AMOUNT GREATER THAN WHAT YOU ACTUALLY PAID FOR THE SPECIFICATION. Should any warranties be found to exist, such warranties shall be limited in duration to ninety (90) days following the date you receive the Specification.

INDEMNIFICATION. By your inclusion of the Paint Shop Pro File Format in your software product(s), you agree to indemnify and hold Corel harmless from any and all claims of any kind or nature made by any of your customers with respect to your software product(s).

EXPORT LAWS. You agree that you and your customers will not export your software or Specification except in compliance with the laws and regulations of the United States.

US GOVERNMENT RESTRICTED RIGHTS. The Specification and any accompanying materials are provided with Restricted Rights. Use, duplication or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of The Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, or subparagraphs (c)(1) and (2) of the Commercial Computer Software - Restricted Rights at 48 CFR 52.227-19, as applicable. Contractor/manufacturer is Corel Corporation, 1600 Carling Avenue, Ottawa, Ontario, KIZ 8R7, Canada.

Corel reserves the right to amend, modify, change, revoke or withdraw the Specification at any time and from time to time. Corel shall have no obligation to support or maintain the Specification.

Table of Contents

1 Version History

Date	Author	Description
April 20, 1998	JMS	Initial draft, submitted for review.
April 23, 1998	JMS	Minor changes.
April 29, 1998	JMS	Minor changes (changed Layer flags, Layer transparency flags to Boolean Layer visible flag, and Transparency protected flag).
August 7, 1998	JMS	Made additions for storing Picture Tube control information in PSP files.
November 16,1998	JCO	Added additional Copyright information, release notes, disclaimers, and comments, and cleaned for public release.
November 23, 1998	JMS	Added restrictions for readers and writers. Changes based on latest review.
November 30, 1998	JCO	Restrictions clarified. Requests email address added.
December 1, 1998	JMS	Added descriptions of compression types. Added descriptions of non-color channels.
January 28, 1999	JMS	• First crack at documenting support for text and vector shape data. • Explicitly stated where we might add fields in the future, which reader should ignore (note that our reader must be updated to ignore this as well!).
February 16, 1999	JMS	Changes after document review.
April 31, 1999	JMS	Updated to PSP file format version 4.0.
September 9, 1999	AS	Initial draft for the new PSP version 4.0 file format. Submitted for review.
September 19, 1999	AS	Changes after document review.
October 20, 1999	AS	Corrected the polyline node type flags, PSPPolylineNodeTypes.
September 11, 2000	AS	Initial draft for the new PSP version 5.0 file format. Submitted for review.
April 24, 2003	AS	Initial draft for the new PSP version 6.0 file format. Submitted for review.
October 20, 2003	SLN	Made updates regarding EXIF makernote storage for PSP 8.1, corrected some ambiguities.

March 5, 2004	SLN	Added new text char style attributes for format version 7 (PSP9).
November 10, 2004	SLN	Added description of art media layers, added picture tube scale factor. Made corrections to Char style definition chunk. Final draft of version 7 format (PSP9).
September 21, 2005	SLN	Updated specification for version 8 format (PSP 10). Added information about color profiles and IPTC information.
September 27, 2005	SLN	Updated restrictions section for 16 bits per channel images. Also minor edits throughout the document for 16 bits per channel support.

2 Introduction

The Paint Shop Pro (PSP) file format is an extremely rich graphics file format that is capable of describing a multi-layered image document in minute detail. The format facilitates the retention of a great deal of information, including layer bitmaps, layer masks, layer vector data, layer adjustment data, layer grouping information, multiple sample thumbnails, multiple sample composite images, image dimension, image resolution, creator name, copyright owner, image description, EXIF information, masks, selections, alpha channels, etc.

The PSP format supports storing image bitmaps in an uncompressed format or, to reduce the size of the file, in either Run Length Encoding (RLE) or LZ77 compressed format. As each of these is a lossless storage method, the PSP format always maintains the original quality of stored image documents.

Finally, the PSP format supports storing image bitmaps at a variety of bit depths, including 48-bit color (16 bits per channel), 24-bit color (16.7 million colors), 8-bit color (256 color), 16 bits per channel greyscale, 8-bit greyscale (256 greys), 4-bit color (16 colors), and 1-bit color (2 colors).

When developing an image document in Paint Shop Pro, the full-service graphics editing tool from Corel, the PSP file format is the format of choice because an image document can be maintained in its entirety. (Saving image documents to less versatile formats results in loss of some image document data.)

This document describes the PSP file format. The next section (Section 3) provides several resources related to the PSP format and Corel products. Section 4 provides a high-level overview of the format. Section 5 provides descriptions of several types, blocks, chunks, and conventions used throughout the remainder of the document. Section 6 provides the details of the PSP file format. Finally, “Appendix A : PSP File Format Overview” provides an overview of the PSP file format. (This overview is more detailed than that of Section 4.)

NOTICE: This document describes the PSP file format, not necessarily the technology or code required to fully read, interpret, or display all aspects of the data that may be contained within the file.

3 Resources

For more information about Corel, Paint Shop Pro, or the PSP file format, please visit the Corel web site and the Paint Shop Pro newsgroup (both listed below). The latest published version of this document is available at the Corel web site. If you have any questions or comments about this document or about the PSP file format, please send them to the Corel PSP File Format email address listed below.

The Corel web site:

http://www.corel.com/

The Paint Shop Pro newsgroup:

comp.graphics.apps.paint-shop-pro

The Corel PSP File Format email address:

PSPFileFormat@corel.com

4 Format Overview

The Paint Shop Pro (PSP) file format is a block-oriented format that consists of a file header with a sequence of up to ten main blocks. The main blocks are the General Image Attributes Block, the Extended Data Block (optional), the Tube Data Block (optional), the Creator Data Block (optional), the Composite Image Bank Block (optional), the Table Bank Block (optional), the Color Palette Block (for paletted images), the Layer Bank Block, the Selection Block (optional), and the Alpha Channel Bank Block (optional). Each of these main blocks contains information about the image document, and some of these blocks contain sub-blocks. See subsequent sections of this document for more detailed information about each of these main blocks.

The following table contains a high-level view of the PSP file format that includes only the header and the top-level blocks.

File Header
General Image Attributes Block
Extended Data Block (optional)
Tube Data Block (optional)
Creator Data Block (optional)
Composite Image Bank Block (optional)
Table Bank Block (optional)
Color Palette Block (paletted image documents only)
Layer Bank Block
Selection Block (optional)
Alpha Bank Block (optional)

Table 1 : PSP Format Overview

Restrictions for PSP format readers:

- PSP format readers must be able to read and successfully process the PSP File Header, the General Image Attributes Block, the Table Bank Block (if present), the Color Palette Block (if present), and the Layer Bank Block. PSP readers may ignore all other main blocks.

- If a PSP reader encounters any unrecognized block (whether documented or not), the reader must ignore (skip over) that block and continue processing the rest of the file. Note that this document describes "expansion fields" throughout the file format. These "expansion fields" mark places in the file where the format has been left open for the addition of additional data. Readers should skip past any data in these "expansion fields." (Note that the "expansion fields" are not undocumented fields; they are currently non-existent fields.)

Restrictions for PSP format writers (Format Versions 6.0 and 7.0):

- PSP format writers must write the PSP File Header, the General Image Attributes Block, the Composite Image Bank Block (if appropriate), the Table Bank Block (if appropriate), the Color Palette Block (if appropriate), and the Layers Bank Block exactly as described in this specification. Additionally, PSP format writers may write the other blocks described in this document when necessary.

- PSP format writers must neither write any blocks, chunks, or chunk fields that are not documented in this specification nor otherwise stray from this specification. Additional block types or specification changes may only be created by Corel. Requests for additional block types and other file format enhancements can be sent to mailto:PSPFileFormat@corel.com.

- PSP format writers must not write any data in the "expansion fields" described in this document. "Expansion fields" are spots in the file that are reserved for future use by Corel.

- Color data of multi raster layer image documents must be stored in either 24-bit truecolor format or in 8-bit greyscale format.

- Color data of single-layer image documents can be stored in 1, 4, or 8-bit paletted format, in 24-bit truecolor format, or in 8-bit greyscale format.

- PSP format writers must not write an image document with more than 500 layers.

Restrictions for PSP format writers (Format Version 8.0):

- Color data of multi raster layer image documents must be stored in either 8 or 16 bits per channel truecolor format or in 8 or 16 bits per channel greyscale format.

- Color data of single-layer image documents can be stored in 1, 4, or 8-bit paletted format, in 8 or 16 bits per channel truecolor format, or in 8 or 16 bits per channel greyscale format.

5 Common Structures and Definitions

This section contains information concerning structures and definitions that are used throughout the remainder of this document. They are presented here to give the user a hint as to what to expect in subsequent sections of the document and to provide references that can be used in subsequent sections of the document to avoid too much repetition (though repetition is used when it provides needed clarification).

5.1 Parlance

With any documentation, it is important to ensure that the author and the audience have a common parlance. With that in mind, the following glossary describes the usage, throughout this document, of various words that otherwise might be ambiguous.

Word	Definition/Usage
Block	As described above, the PSP format is a block-oriented format. The term "block" will be used to refer to a part of a PSP file that consists of a Block Header (See Section "5.4.1 - Block Header") ***and*** all the data following the header that is associated with the header. Since the Block Header indicates the size of the block, it is quickly apparent how much of the data immediately following the block header is associated with it.
Chunk	The term chunk will be used to represent an "interesting" piece of data within a PSP file that does not have a block header. Throughout the document, there is a need to refer to a piece of data within a block that is not a sub-block. The term "chunk" is used in these cases. It is worthwhile to note that the chunk length field of each chunk includes the length field in the length calculation. This is different than blocks, since the block header is not included in the block length calculation.
Main block	The PSP file format is a hierarchical format, with some blocks that are fully contained within other blocks. A main block is any block in a PSP file that is not contained within another block. The main blocks are illustrated in the table in Section 4.
Sub-Block	A sub-block is a block that is fully contained within another block.

Table 2: Glossary of Terms

5.2 Type Definitions

The following table contains definitions for simple data types used throughout this document. The PSP file format is always written in Intel® (or little-endian) byte order.

Type	Description
BYTE	An 8-bit unsigned integer value.
B_ARRAY	An array of 8-bit unsigned integer values (i.e., an array of BYTE). The length of the array is either fixed or can be determined from the discussion.
C_ARRAY	An array of 8-bit signed integer values (i.e., an array of char). The length of the array is either fixed or can be determined from the discussion.
DOUBLE	A 64-bit floating point value.
DWORD	A 32-bit unsigned integer value.

LONG	A 32-bit signed integer value.
RECT	An array of 4 LONGs that describe a rectangle. (The first two LONGs are the x-coordinate and y-coordinate (respectively) of the upper-left corner of the rectangle; the third and fourth LONGs are the x-coordinate and y-coordinate (respectively) of the lower-right corner of the rectangle.)
RGBQUAD	An array of 4 BYTEs representing a color palette entry, where the first BYTE is the entry's red component, the second BYTE is the entry's green component, the third BYTE is the entry's blue component, and the fourth BYTE is always 0 (reserved).
WORD	A 16-bit unsigned integer value.
COLOR	The COLOR data type is used to represent a color. Its definition is based on whether the PSP file contains a paletted image: • If the PSP file contains a paletted image, the COLOR data type is a one BYTE index into the image's color palette (note that a global color palette is used). • If the PSP file contains a 24-bit truecolor image, the COLOR data type is composed of three bytes, containing the red, green, and blue components of the represented color.

Table 3: Common type definitions

5.3 Limiting Field Values

The following enumerations are used throughout the document to limit the allowed values of various fields. As an example, the Block Identifier field in a Block Header is a DWORD, but, as detailed in the Block Identifier field description, the DWORD must be in the range covered by the enumeration PSPBlockID.

```
/* Block identifiers.
 */
enum PSPBlockID
{
      PSP_IMAGE_BLOCK = 0,    // General Image Attributes Block (main)
      PSP_CREATOR_BLOCK,      // Creator Data Block (main)
      PSP_COLOR_BLOCK,        // Color Palette Block (main and sub)
      PSP_LAYER_START_BLOCK,  // Layer Bank Block (main)
      PSP_LAYER_BLOCK,        // Layer Block (sub)
      PSP_CHANNEL_BLOCK,      // Channel Block (sub)
      PSP_SELECTION_BLOCK,    // Selection Block (main)
      PSP_ALPHA_BANK_BLOCK,   // Alpha Bank Block (main)
      PSP_ALPHA_CHANNEL_BLOCK,// Alpha Channel Block (sub)
      PSP_COMPOSITE_IMAGE_BLOCK, // Composite Image Block (sub)
      PSP_EXTENDED_DATA_BLOCK,// Extended Data Block (main)
      PSP_TUBE_BLOCK,         // Picture Tube Data Block (main)
      PSP_ADJUSTMENT_EXTENSION_BLOCK, // Adjustment Layer Block (sub)
      PSP_VECTOR_EXTENSION_BLOCK, // Vector Layer Block (sub)
      PSP_SHAPE_BLOCK,        // Vector Shape Block (sub)
      PSP_PAINTSTYLE_BLOCK,   // Paint Style Block (sub)
      PSP_COMPOSITE_IMAGE_BANK_BLOCK, // Composite Image Bank (main)
      PSP_COMPOSITE_ATTRIBUTES_BLOCK, // Composite Image Attr. (sub)
      PSP_JPEG_BLOCK,         // JPEG Image Block (sub)
      PSP_LINESTYLE_BLOCK,    // Line Style Block (sub)
      PSP_TABLE_BANK_BLOCK,   // Table Bank Block (main)
      PSP_TABLE_BLOCK,        // Table Block (sub)
      PSP_PAPER_BLOCK,        // Vector Table Paper Block (sub)
      PSP_PATTERN_BLOCK,      // Vector Table Pattern Block (sub)
      PSP_GRADIENT_BLOCK,     // Vector Table Gradient Block (not used)
      PSP_GROUP_EXTENSION_BLOCK, // Group Layer Block (sub)
      PSP_MASK_EXTENSION_BLOCK, // Mask Layer Block (sub)

      PSP_BRUSH_BLOCK,         // Brush Data Block (main)
```

```
      PSP_ART_MEDIA_BLOCK,     // Art Media Layer Block (main)
      PSP_ART_MEDIA_MAP_BLOCK, // Art Media Layer map data Block (main)
      PSP_ART_MEDIA_TILE_BLOCK, // Art Media Layer map tile Block(main)
      PSP_ART_MEDIA_TEXTURE_BLOCK,// AM Layer map texture Block (main)

      PSP_COLORPROFILE_BLOCK, // ICC Color profile block
};


/* Truth values.
 */
enum PSP_BOOLEAN
{
      FALSE = 0,
      TRUE,
};


/* Possible metrics used to measure resolution.
 */
enum PSP_METRIC
{
      PSP_METRIC_UNDEFINED = 0,      // Metric unknown
      PSP_METRIC_INCH,               // Resolution is in inches
      PSP_METRIC_CM,                 // Resolution is in centimeters
};


/* Creator application identifiers.
 */
enum PSPCreatorAppID
{
      PSP_CREATOR_APP_UNKNOWN = 0,     // Creator application unknown
      PSP_CREATOR_APP_PAINT_SHOP_PRO,  // Creator is Paint Shop Pro
};


/* Creator field types.
 */
enum PSPCreatorFieldID
{
      PSP_CRTR_FLD_TITLE = 0,        // Image document title field
      PSP_CRTR_FLD_CRT_DATE,         // Creation date field
      PSP_CRTR_FLD_MOD_DATE,         // Modification date field
      PSP_CRTR_FLD_ARTIST,           // Artist name field
      PSP_CRTR_FLD_CPYRGHT,          // Copyright holder name field
      PSP_CRTR_FLD_DESC,             // Image document description field
      PSP_CRTR_FLD_APP_ID,           // Creating app id field
      PSP_CRTR_FLD_APP_VER,          // Creating app version field
};


/* Extended data field types.
 */
enum PSPExtendedDataID
{
      PSP_XDATA_TRNS_INDEX = 0,      // Transparency index field
      PSP_XDATA_GRID,                // Image grid information
      PSP_XDATA_GUIDE,               // Image guide information
      PSP_XDATA_EXIF,                // Image EXIF information
      PSP_XDATA_IPTC,                // Image IPTC information
};
```

```
/* Grid units type.
 */
enum PSPGridUnitsType
{
      keGridUnitsPixels = 0,  // Grid units is pixels
      keGridUnitsInches,      // Grid units is inches
      keGridUnitsCentimeters  // Grid units is centimeters
};


/* Guide orientation type.
 */
enum PSPGuideOrientationType
{
      keHorizontalGuide = 0,  // Horizontal guide direction
      keVerticalGuide         // Vertical guide direction
};


/* Bitmap types.
 */
enum PSPDIBType
{
      PSP_DIB_IMAGE = 0,            // Layer color bitmap
      PSP_DIB_TRANS_MASK,           // Layer transparency mask bitmap
      PSP_DIB_USER_MASK,            // Layer user mask bitmap
      PSP_DIB_SELECTION,            // Selection mask bitmap
      PSP_DIB_ALPHA_MASK,           // Alpha channel mask bitmap
      PSP_DIB_THUMBNAIL,            // Thumbnail bitmap
      PSP_DIB_THUMBNAIL_TRANS_MASK, // Thumbnail transparency mask
      PSP_DIB_ADJUSTMENT_LAYER,     // Adjustment layer bitmap
      PSP_DIB_COMPOSITE,            // Composite image bitmap
      PSP_DIB_COMPOSITE_TRANS_MASK, // Composite image transparency
      PSP_DIB_PAPER,                // Paper bitmap
      PSP_DIB_PATTERN,              // Pattern bitmap
      PSP_DIB_PATTERN_TRANS_MASK,   // Pattern transparency mask
};


/* Type of image in the composite image bank block.
 */
enum PSPCompositeImageType
{
    PSP_IMAGE_COMPOSITE = 0,  // Composite Image
    PSP_IMAGE_THUMBNAIL,      // Thumbnail Image
};


/* Channel types.
 */
enum PSPChannelType
{
      PSP_CHANNEL_COMPOSITE = 0,    // Channel of single channel bitmap
      PSP_CHANNEL_RED,              // Red channel of 8, 16 bpc bitmap
      PSP_CHANNEL_GREEN,            // Green channel of 8,16 bpc bitmap
      PSP_CHANNEL_BLUE,             // Blue channel of 8, 16 bpc bitmap
};


/* Possible types of compression.
 */
enum PSPCompression
{
```

```
      PSP_COMP_NONE = 0,            // No compression
      PSP_COMP_RLE,                 // RLE compression
      PSP_COMP_LZ77,                // LZ77 compression
      PSP_COMP_JPEG                 // JPEG compression (only used by
                                    // thumbnail and composite image)
};


/* Layer types.
 */
enum PSPLayerType
{
      keGLTUndefined = 0,           // Undefined layer type
      keGLTRaster,                  // Standard raster layer
      keGLTFloatingRasterSelection, // Floating selection (raster)
      keGLTVector,                  // Vector layer
      keGLTAdjustment,              // Adjustment layer
      keGLTGroup,                   // Group layer
      keGLTMask,                    // Mask layer
      keGLTArtMedia                 // Art media layer
};


/* Layer flags.
 */
enum PSPLayerProperties
{
      keVisibleFlag           = 0x00000001,     // Layer is visible
      keMaskPresenceFlag      = 0x00000002,     // Layer has a mask
};


/* Blend modes.
 */
enum PSPBlendModes
{
      LAYER_BLEND_NORMAL,
      LAYER_BLEND_DARKEN,
      LAYER_BLEND_LIGHTEN,
      LAYER_BLEND_LEGACY_HUE,
      LAYER_BLEND_LEGACY_SATURATION,
      LAYER_BLEND_LEGACY_COLOR,
      LAYER_BLEND_LEGACY_LUMINOSITY,
      LAYER_BLEND_MULTIPLY,
      LAYER_BLEND_SCREEN,
      LAYER_BLEND_DISSOLVE,
      LAYER_BLEND_OVERLAY,
      LAYER_BLEND_HARD_LIGHT,
      LAYER_BLEND_SOFT_LIGHT,
      LAYER_BLEND_DIFFERENCE,
      LAYER_BLEND_DODGE,
      LAYER_BLEND_BURN,
      LAYER_BLEND_EXCLUSION,
      LAYER_BLEND_TRUE_HUE,
      LAYER_BLEND_TRUE_SATURATION,
      LAYER_BLEND_TRUE_COLOR,
      LAYER_BLEND_TRUE_LIGHTNESS,
      LAYER_BLEND_ADJUST = 255,
};


/* Adjustment layer types.
 */
enum PSPAdjustmentLayerType
```

```
{
      keAdjNone = 0,          // Undefined adjustment layer type
      keAdjLevel,             // Level adjustment
      keAdjCurve,             // Curve adjustment
      keAdjBrightContrast,    // Brightness-contrast adjustment
      keAdjColorBal,          // Color balance adjustment
      keAdjHSL,               // HSL adjustment
      keAdjChannelMixer,      // Channel mixer adjustment
      keAdjInvert,            // Invert adjustment
      keAdjThreshold,         // Threshold adjustment
      keAdjPoster             // Posterize adjustment
};

/* Art media layer map types
 */
enum PSPArtMediaMapType
{
    keArtMediaColorMap = 0,
    keArtMediaBumpMap,
    keArtMediaShininessMap,
    keArtMediaReflectivityMap,
    keArtMediaDrynessMap
};


/* Vector shape types.
 */
enum PSPVectorShapeType
{
      keVSTUnknown = 0, // Undefined vector type
      keVSTText,        // Shape represents lines of text
      keVSTPolyline,    // Shape represents a multiple segment line
      keVSTEllipse,     // Shape represents an ellipse (or circle)
      keVSTPolygon,     // Shape represents a closed polygon
      keVSTGroup,       // Shape represents a group shape
};


/* Shape property flags
 */
enum PSPShapeProperties
{
      keShapeAntiAliased      = 0x00000001,  // Shape is anti-aliased
      keShapeSelected         = 0x00000002,  // Shape is selected
      keShapeVisible          = 0x00000004,  // Shape is visible
};


/* Polyline node type flags.
 */
enum PSPPolylineNodeTypes
{
      keNodeUnconstrained     = 0x0000, // Default node type
      keNodeSmooth            = 0x0001, // Node is smooth
      keNodeSymmetric         = 0x0002, // Node is symmetric
      keNodeAligned           = 0x0004, // Node is aligned
      keNodeActive            = 0x0008, // Node is active
      keNodeLocked            = 0x0010, // Node is locked
      keNodeSelected          = 0x0020, // Node is selected
      keNodeVisible           = 0x0040, // Node is visible
      keNodeClosed            = 0x0080, // Node is closed
};
```

```
/* Paint style types.
 */
enum PSPPaintStyleType
{
      keStyleNone       = 0x0000,  // No paint style info applies
      keStyleColor      = 0x0001,  // Color paint style info
      keStyleGradient   = 0x0002,  // Gradient paint style info
      keStylePattern    = 0x0004,  // Pattern paint style info
      keStylePaper      = 0x0008,  // Paper paint style info
      keStylePen        = 0x0010,  // Organic pen paint style info
};


/* Gradient type.
 */
enum PSPStyleGradientType
{
      keSGTLinear = 0,  // Linera gradient type
      keSGTRadial,      // Radial gradient type
      keSGTRectangular, // Rectangulat gradient type
      keSGTSunburst     // Sunburst gradient type
};


/* Paint Style Cap Type (Start & End).
 */
enum PSPStyleCapType
{
      keSCTCapFlat = 0,       // Flat cap type (was round in psp6)
      keSCTCapRound,          // Round cap type (was square in psp6)
      keSCTCapSquare,         // Square cap type (was flat in psp6)
      keSCTCapArrow,          // Arrow cap type
      keSCTCapCadArrow,       // Cad arrow cap type
      keSCTCapCurvedTipArrow, // Curved tip arrow cap type
      keSCTCapRingBaseArrow,  // Ring base arrow cap type
      keSCTCapFluerDelis,     // Fluer deLis cap type
      keSCTCapFootball,       // Football cap type
      keSCTCapXr71Arrow,      // Xr71 arrow cap type
      keSCTCapLilly,          // Lilly cap type
      keSCTCapPinapple,       // Pinapple cap type
      keSCTCapBall,           // Ball cap type
      keSCTCapTulip           // Tulip cap type
};


/* Paint Style Join Type.
 */
enum PSPStyleJoinType
{
      keSJTJoinMiter = 0,     // Miter join type
      keSJTJoinRound,         // Round join type
      keSJTJoinBevel          // Bevel join type
};


/* Organic pen type.
 */
enum PSPStylePenType
{
      keSPTOrganicPenNone = 0,      // Undefined pen type
      keSPTOrganicPenMesh,          // Mesh pen type
      keSPTOrganicPenSand,          // Sand pen type
      keSPTOrganicPenCurlicues,     // Curlicues pen type
      keSPTOrganicPenRays,          // Rays pen type
```

```
      keSPTOrganicPenRipple,        // Ripple pen type
      keSPTOrganicPenWave,          // Wave pen type
      keSPTOrganicPen               // Generic pen type
};

/* Text element types.
*/
enum PSPTextElementType
{
      keTextElemUnknown = 0,  // Undefined text element type
      keTextElemChar,         // A single character code
      keTextElemCharStyle,    // A character style change
      keTextElemLineStyle     // A line style change
};


/* Text alignment types.
 */
enum PSPTextAlignment
{
      keTextAlignmentLeft = 0,      // Left text alignment
      keTextAlignmentCenter,        // Center text alignment
      keTextAlignmentRight          // Right text alignment
};

/* Text antialias modes.
 */
enum PSPAntialiasMode
{
      keNoAntialias = 0,            // Antialias off
      keSharpAntialias,             // Sharp
      keSmoothAntialias             // Smooth
};




/* Text flow types
 */
enum PSPTextFlow
{
      keTFHorizontalDown = 0,       // Horizontal then down
      keTFVerticalLeft,             // Vertical then left
      keTFVerticalRight,            // Vertical then right
      keTFHorizontalUp              // Horizontal then up
};


/* Character style flags.
 */
enum PSPCharacterProperties
{
      keStyleItalic     = 0x00000001,     // Italic property bit
      keStyleStruck     = 0x00000002,     // Strike-out property bit
      keStyleUnderlined = 0x00000004,     // Underlined property bit
      keStyleWarped     = 0x00000008,     // Warped property bit
      keStyleAntiAliased = 0x00000010,    // Anti-aliased property bit
};


/* Table type.
```

```
 */
enum PSPTableType
{
      keTTUndefined = 0,      // Undefined table type
      keTTGradientTable,      // Gradient table type
      keTTPaperTable,         // Paper table type
      keTTPatternTable        // Pattern table type
};


/* Picture tube placement mode.
 */
enum TubePlacementMode
{
      tpmRandom,        // Place tube images in random intervals
      tpmConstant,      // Place tube images in constant intervals
};


/* Picture tube selection mode.
 */
enum TubeSelectionMode
{
      tsmRandom,        // Randomly select the next image in
                        // tube to display
      tsmIncremental,   // Select each tube image in turn
      tsmAngular,       // Select image based on cursor direction
      tsmPressure,      // Select image based on pressure
                        // (from pressure-sensitive pad)
      tsmVelocity,      // Select image based on cursor speed
};


/* Graphic contents flags.
 */
enum PSPGraphicContents
{
      // Layer types
      keGCRasterLayers      = 0x00000001, // At least one raster layer
      keGCVectorLayers      = 0x00000002, // At least one vector layer
      keGCAdjustmentLayers  = 0x00000004, // At least one adjust. layer
      keGCGroupLayers       = 0x00000008, // at least one group layer
      keGCMaskLayers        = 0x00000010, // at least one mask layer
      keGCArtMediaLayers = 0x00000020, // at least one art media layer

      // Additional attributes
      keGCMergedCache = 0x00800000, // merged cache (composite image)
      keGCThumbnail              = 0x01000000, // Has a thumbnail
      keGCThumbnailTransparency  = 0x02000000, // Thumbnail transp.
      keGCComposite              = 0x04000000, // Has a composite image
      keGCCompositeTransparency  = 0x08000000, // Composite transp.
      keGCFlatImage              = 0x10000000, // Just a background
      keGCSelection              = 0x20000000, // Has a selection
      keGCFloatingSelectionLayer = 0x40000000, // Has float. selection
      keGCAlphaChannels          = 0x80000000, // Has alpha channel(s)
};
```

5.4 Common Blocks and Common Chunks

This section describes various blocks and chunks that are used throughout the remainder of the document. Recall that a “block” includes a PSP Block Header (see below) and all data related to the header, whereas a “chunk” is simply a group of data that is related (but has no header).

These blocks and chunks are described here in order to simplify reference to them later in the document (either because they are referenced often or because it is easier to refer the reader to the definitions here than to embed their descriptions within the context of their use).

5.4.1 Block Header

The block structure utilized by the PSP file format is quite simple. Each main block consists of a Block Header, which indicates size and type information for the block, followed by the block data. The content of the block data depends on the block's type.

Moreover, the block structure used in the PSP format is hierarchical; a block may consist of one or more sub-blocks. The sub-blocks take the same format as the main blocks (block header + block data).

As illustrated in the following table, the Block Header identifies the block's type and the total block length.

Type	Length	Name	Description
B_ARRAY	4 bytes	Header identifier	Always "`7E 42 4B 00`" (i.e.,"~BK" followed by a zero byte).
WORD	2 bytes	Block identifier	Identifies the type of the block (one of PSPBlockID).
DWORD	4 bytes	Total block length	Total length of block (including all sub-blocks), not including the header. Use this length to skip past unknown or unwanted blocks.

5.4.2 Variable Length String Chunk

Variable length strings are used in the PSP format to store textual data, such as a layer name. This section describes the representation of variable length strings in the PSP format.

As illustrated in the following table, the Variable Length String Chunk consists of a character count and the array of characters that comprise the variable length string. The Variable Length String Chunk does not contain the string's ending NULL character.

Type	Length	Name	Description
WORD	2 bytes	Character count	Count of following characters (let's call this j).
C_ARRAY	j bytes	Characters	Characters that comprise the string.

5.4.3 Color Palette Sub-Block

Color Palette Sub-Blocks are used throughout the PSP file format to store palettes associated with paletted bitmaps. For example, when a composite image is stored as a paletted bitmap, there is a Color Palette Sub-Block in the Composite Image Sub-Block to store the composite image's palette.

As illustrated in the following table, the Color Palette Sub-Block consists of a Color Palette Block Header, a Color Palette Information Chunk, and a Color Palette Entries Chunk.

Color Palette Block Header:

Type	Length	Name	Description
B_ARRAY	4 bytes	Header identifier	Always "7E 42 4B 00" (i.e.,"~BK", followed by a zero byte).
WORD	2 bytes	Block identifier	Always PSP_COLOR_BLOCK.
DWORD	4 bytes	Total block length	Length of complete Color Palette Sub-Block, including Color Palette Information Chunk and Color Palette Entries Chunk.

Color Palette Information Chunk:

DWORD	4 bytes	Chunk size	Length of Color Palette Information Chunk.
DWORD	4 bytes	Palette entry count	Number of entries in the palette (for the purpose of determining the number of fields in the Color Palette Entries Chunk, let's call this i).
	unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Color Palette Entries Chunk

RGBQUAD	4 bytes	Palette entry 1	The 1st palette entry.
…	…	…	…
RGBQUAD	4 bytes	Palette entry i	The ith palette entry

5.4.4 Paint Style Sub-Block

The Paint Style Sub-Block is used to describe how an item is to be painted. For example, it is used to describe how each vector shape's outline and fill should be painted in the Vector Shape Sub-Block.

As illustrated below, the Paint Style Sub-Block consists of the Paint Style Block Header, the Paint Style Information Chunk, and the Paint Style Definition Chunks.

Paint Style Block Header:

Type	Length	Name	Description
B_ARRAY	4 bytes	Header identifier	Always "7E 42 4B 00" (i.e.,"~BK", followed by a zero byte).
WORD	2 bytes	Block identifier	Always PSP_PAINTSTYLE_BLOCK.

DWORD	4 bytes	Total block length	Length of the complete Paint Style Sub-Block, including the Paint Style Information Chunk and all the Paint Style Definition Chunks.

Paint Style Information Chunk:

DWORD	4 bytes	Chunk size	Length of Paint Style Information Chunk.
WORD	2 bytes	Paint style type flags	A series of flags (in PSPPaintStyleType) that defines the paint style type(s).
	unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Paint Style Definition Chunks:

The contents of the Paint Style Definition Chunks are based on the paint style type flags defined in the previous Paint Style Information Chunk. The Color Paint Style Definition Chunk is present if the keStyleColor paint style type flag is defined. The Gradient Paint Style Definition Chunk is present if the keStyleGradient paint style type flag is defined. The Pattern Paint Style Definition Chunk is present if the keStylePattern paint style type flag is defined. The Paper Paint Style Definition Chunk is present if the keStylePaper paint style type flag is defined. The Organic Pen Paint Style Definition Chunk is present if the keStylePen paint style type flag is defined.

Color Paint Style Definition Chunk:

DWORD	4 bytes	Chunk size	Length of Color Paint Style Definition Chunk.
DWORD	4 bytes	RGB color definition	RGB color definition
LONG	4 bytes	Color palette index	Color palette index (-1 if not paletted)
	unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Gradient Paint Style Definition Chunk:

DWORD	4 bytes	Chunk size	Length of Gradient Paint Style Definition Chunk.
Variable length string chunk	See Section 5.4.2	Gradient name	Gradient name. (Please See Section "5.4.2 – Variable Length String Chunk" for more information about variable string chunks.)
LONG	4 bytes	Gradient identifier	Table entry identifier reserved for future implementation of gradient table (currently -1).

BYTE	1 byte	Invert flag	Indicates whether the gradient is inverted (0 = not inverted, 1 = inverted).
LONG	4 bytes	Center point horizontal	Horizontal (x) value of gradient center point.
LONG	4 bytes	Center point vertical	Vertical (y) value of gradient center point.
LONG	4 bytes	Focal point horizontal	Horizontal (x) value of gradient focal point.
LONG	4 bytes	Focal point vertical	Vertical (y) value of gradient focal point.
DOUBLE	8 bytes	Angle	Gradient angle (for linear gradients only).
WORD	2 bytes	Repeats	Number of repeats of the gradient.
WORD	2 bytes	Type	Type of gradient (must be one of PSPStyleGradientType).
WORD	2 bytes	Color count	Number of Gradient Color Chunk(s) following.
WORD	2 bytes	Transparency count	Number of Gradient Transparency Chunk(s) following.
	unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Gradient Color Chunk (first color):

DWORD	4 bytes	Chunk size	Length of Gradient Color Chunk.
DWORD	4 bytes	Color	Gradient color (RGB).
WORD	2 bytes	Location	Gradient color location.
WORD	2 bytes	Midpoint	Gradient color midpoint location (from previous color).
	unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

• • •

Gradient Color Chunk (last color):

DWORD	4 bytes	Chunk size	Length of Gradient Color Chunk.
DWORD	4 bytes	Color	Gradient color (RGB).
WORD	2 bytes	Location	Gradient color location.
WORD	2 bytes	Midpoint	Gradient color midpoint location (from previous color).
	unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Gradient Transparency Chunk (first transparency):

DWORD	4 bytes	Chunk size	Length of Gradient Transparency Chunk.
WORD	2 bytes	Opacity	Gradient transparency opacity (0 is fully transparent - 100 is fully opaque).
WORD	2 bytes	Location	Gradient transparency location.
WORD	2 bytes	Midpoint	Gradient transparency midpoint location (from previous transparency).
	unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

• • •

Gradient Transparency Chunk (last transparency):

DWORD	4 bytes	Chunk size	Length of Gradient Transparency Chunk.
WORD	2 bytes	Opacity	Gradient transparency opacity (0 is fully transparent - 100 is fully opaque).
WORD	2 bytes	Location	Gradient transparency location.
WORD	2 bytes	Midpoint	Gradient transparency midpoint location (from previous transparency).
	unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Pattern Paint Style Definition Chunk:

DWORD	4 bytes	Chunk size	Length of Pattern Paint Style Definition Chunk.
Variable length string chunk	See Section 5.4.2	Pattern name	Pattern name. (Please See Section "5.4.2 – Variable Length String Chunk" for more information about variable string chunks.)
LONG	4 bytes	Pattern identifier	Pattern table entry identifier. This is an index into the pattern table defined in the Table Bank Block. (Please See Section "6.9 –The Table Bank Block" for more information about tables.)
DOUBLE	8 bytes	Rotation	Pattern rotation angle.
DOUBLE	8 bytes	Scale	Pattern scale factor.
BYTE	1 byte	Flip flag	Indicates whether the pattern is flipped (0 = not flipped, 1 = flipped).
BYTE	1 byte	Mirror flag	Indicates whether the pattern is mirrored (0 = not mirrored, 1 = mirrored).
	unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Paper Paint Style Definition Chunk:

DWORD	4 bytes	Chunk size	Length of Paper Paint Style Definition Chunk.
Variable length string chunk	See Section 5.4.2	Paper name	Paper name. (Please See Section "5.4.2 – Variable Length String Chunk" for more information about variable string chunks.)
LONG	4 bytes	Paper identifier	Paper table entry identifier. This is an index into the paper table defined in the Table Bank Block. (Please See Section "6.9 –The Table Bank Block" for more information about tables.)
DOUBLE	8 bytes	Rotation	Paper rotation angle.
DOUBLE	8 bytes	Scale	Paper scale factor.
BYTE	1 byte	Flip flag	Indicates whether the paper is flipped (0 = not flipped, 1 = flipped).
BYTE	1 byte	Mirror flag	Indicates whether the paper is mirrored (0 = not mirrored, 1 = mirrored).
	unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Organic Pen Paint Style Definition Chunk:

DWORD	4 bytes	Chunk size	Length of Organic Pen Paint Style Definition Chunk.
LONG	4 bytes	Pen identifier	Table entry identifier reserved for future implementation of pen table (currently -1).
WORD	2 bytes	Type	Organic pen type (must be one of PSPStylePenType).
DWORD	4 bytes	First start color	First start color (RGB).
DWORD	4 bytes	First end color	First end color (RGB).
DWORD	4 bytes	Second start color	Second start color (RGB).
DWORD	4 bytes	Second end color	Second end color (RGB).
BYTE	1 byte	Key effect flag	Indicates whether to use the key effect (0 = don't use key effect, 1 = use key effect).
BYTE	1 byte	Extra color flag	Indicates whether to use the extra (second) color (0 = don't use extra color, 1 = use extra color).
LONG	4 bytes	Parameter 1	Organic pen parameter 1.
LONG	4 bytes	Parameter 2	Organic pen parameter 2.
LONG	4 bytes	Parameter 3	Organic pen parameter 3.
LONG	4 bytes	Parameter 4	Organic pen parameter 4.

LONG	4 bytes	Parameter 5	Organic pen parameter 5.
LONG	4 bytes	Parameter 6	Organic pen parameter 6.
LONG	4 bytes	Parameter 7	Organic pen parameter 7.
LONG	4 bytes	Parameter 8	Organic pen parameter 8.
LONG	4 bytes	Parameter 9	Organic pen parameter 9.
LONG	4 bytes	Parameter 10	Organic pen parameter 10.
	unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

5.4.5 Line Style Sub-Block

The Line Style Sub-Block contains styled line data used by the outline paint style of vector shapes in the Vector Shape Sub-Block.

As illustrated below, the Line Style Sub-Block consists of the Line Style Block Header, the Line Style Information Chunk, and the Line Style Entries Chunk.

Line Style Block Header:

Type	Length	Name	Description
B_ARRAY	4 bytes	Header identifier	Always "`7E 42 4B 00`" (i.e.,"~BK", followed by a zero byte).
WORD	2 bytes	Block identifier	Always PSP_LINESTYLE_BLOCK.
DWORD	4 bytes	Total block length	Length of the complete Line Style Sub-Block, including the Line Style Information Chunk and the Line Style Entries Chunk.

Line Style Information Chunk:

DWORD	4 bytes	Chunk size	Length of Line Style Information Chunk.
BYTE	1 byte	Start cap type	Segment start cap type (must be one of PSPStyleCapType).
BYTE	1 byte	Start cap multipliers flag	TRUE if use the following segment start cap multipliers, FALSE otherwise.
DOUBLE	8 bytes	Start cap width multiplier	Segment start cap width (x) multiplier.
DOUBLE	8 bytes	Start cap height multiplier	Segment start cap height (y) multiplier.
BYTE	1 byte	End cap type	Segment end cap type (must be one of PSPStyleCapType).

BYTE	1 byte	End cap multipliers flag	TRUE if use the following segment end cap multipliers, FALSE otherwise.
DOUBLE	8 bytes	End cap width multiplier	Segment end cap width (x) multiplier.
DOUBLE	8 bytes	End cap height multiplier	Segment end cap height (y) multiplier.
BYTE	1 byte	Link caps flag	TRUE if the line segment caps are linked to the shape's stroke outline caps, FALSE otherwise.
DWORD	4 bytes	Dash-gap count	Number of dash-gap entries in the following Line Style Entries Chunk.
	unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Line Style Entries Chunk:

DWORD	4 bytes	First dash-gap entry	The first dash-gap value of the styled line.
…	…	…	…
DWORD	4 bytes	Last dash-gap entry	The last dash-gap value of the styled line.

5.4.6 Composite Image Attributes Sub-Block

The Composite Image Attributes Sub-Block contains information about a composite image or a thumbnail stored within the PSP file.

As illustrated in the table below, the Composite Image Attributes Sub-Block consists of a Composite Image Attributes Block Header and a Composite Image Attributes Information Chunk.

Composite Image Attributes Block Header:

Type	Length	Name	Description
B_ARRAY	4 bytes	Header identifier	Always "`7E 42 4B 00`" (i.e.,"~BK", followed by a zero byte).
WORD	2 bytes	Block identifier	Always PSP_COMPOSITE_ATTRIBUTES_BLOCK.
DWORD	4 bytes	Total block length	Length of complete Composite Image Attributes Sub-Block, including the Composite Image Attributes Information Chunk.

Composite Image Attributes Information Chunk:

DWORD	4 bytes	Chunk size	Length of Composite Image Attributes Information Chunk.
LONG	4 bytes	Width	Specifies the width of the composite image, in pixels.
LONG	4 bytes	Height	Specifies the height of the composite image, in pixels.

WORD	2 bytes	Bit depth	Number of bits used to represent each color pixel of the composite image (must be 1, 4, 8, 24, or 48).
WORD	2 bytes	Compression type	Type of compression used to compress the composite image (one of PSPCompression, including PSP_COMP_JPEG).
WORD	2 byte	Plane count	Number of planes in the composite image (this value must be 1).
DWORD	4 bytes	Color count	Number of colors in the image ($2^{\text{Bit depth}}$).
WORD	2 bytes	Composite image type	Type of composite image (one of PSPCompositeImageType).
	unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

5.4.7 Composite Image Sub-Block

The Composite Image Sub-Block contains a composite image or a thumbnail in the PSP file. The contents of this sub-block are based on the composite image compression type.

If the compression type is PSP_COMP_JPEG, the contents are as described in the JPEG Composite Image Sub-Block (below).

Otherwise (if the compression type is not PSP_COMP_JPEG), the contents are as described in the Normal Composite Image Sub-Block (below).

5.4.7.1 JPEG Composite Image Sub-Block

The JPEG Composite Image Sub-Block contains a JPEG compressed composite image or thumbnail in the PSP file.

As illustrated in the table below, the JPEG Composite Image Sub-Block consists of a JPEG Block Header, a JPEG Information Chunk, and a JPEG Content Chunk.

JPEG Block Header:

Type	Length	Name	Description
B_ARRAY	4 bytes	Header identifier	Always "`7E 42 4B 00`" (i.e.,"~BK", followed by a zero byte).
WORD	2 bytes	Block identifier	Always PSP_ JPEG_BLOCK.
DWORD	4 bytes	Total block length	Length of complete JPEG Composite Image Sub-Block, including JPEG Information Chunk and JPEG Content Chunk.

JPEG Information Chunk:

DWORD	4 bytes	Chunk size	Length of JPEG Information Chunk.

DWORD	4 bytes	Compressed image size	Size of the image in JPEG compressed form (for the purpose of determining the length of the last field in this block, let's call this size j).
DWORD	4 bytes	Uncompressed image size	Size of the image in uncompressed form.
WORD	2 bytes	Image type	Type of image (one of PSPDIBType).
	Unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

JPEG Content Chunk:

B_ARRAY	j bytes	JPEG image content	This field contains JPEG compressed data that defines the image.

5.4.7.2 Normal Composite Image Sub-Block

The Normal Composite Image Sub-Block contains a composite image or a thumbnail in the PSP file.

As illustrated in the table below, the Normal Composite Image Sub-Block consists of a Composite Image Block Header, a Composite Image Information Chunk, a Composite Image Color Palette Sub-Block (if any), and the Composite Image Channel Sub-Blocks.

Composite Image Block Header:

Type	Length	Name	Description
B_ARRAY	4 bytes	Header identifier	Always "`7E 42 4B 00`" (i.e.,"~BK", followed by a zero byte).
WORD	2 bytes	Block identifier	Always PSP_COMPOSITE_IMAGE_BLOCK.
DWORD	4 bytes	Total block length	Length of complete Composite Image Block, including Composite Image Information Chunk, Composite Image Color Palette Sub-Block (if any), and all Composite Image Channel Sub-Blocks.

Composite Image Information Chunk:

DWORD	4 bytes	Chunk size	Length of Composite Image Information Chunk.
WORD	2 bytes	Count of bitmaps	Number of bitmaps to follow (1 if the composite image has no transparency mask; 2 if the composite image has a transparency mask).
WORD	2 bytes	Count of channels	Number of channels to follow.
	Unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Composite Image Color Palette Sub-Block:

Color Palette Sub-Block containing the composite image's color palette (this Sub-Block is not present in 24-bit and 48-bit images). See Section "5.4.3 : Color Palette Sub-Block."

Composite Image Channel Sub-Blocks:

All the composite image channels. Possible channels include: 1) One or three Channel Sub-Blocks defining the composite image bitmap. If the composite image is a paletted bitmap, there is one channel; if the composite image is a 24-bit or 48-bit bitmap, there are three channels. 2) One channel defining the composite image transparency mask. The transparency mask is stored as an 8-bit greyscale bitmap. See Section "5.4.8: Channel Sub-Block and Channel Compression."

5.4.8 Channel Sub-Block and Channel Compression

Channel Sub-Blocks are used throughout the PSP file format to store image document channel data. Each color channel is stored separately in the PSP format. For example, when a composite image is stored as a 24-bit bitmap, there are 3 Channel Sub-Blocks (a red Channel Sub-Block, a green Channel Sub-Block, and a blue Channel Sub-Block) in the Composite Image Sub-Block. However, when a composite image is stored as a paletted bitmap, there will only be one Channel Sub-Block in the Composite Image Sub-Block (because a paletted bitmap contains only one channel).

As illustrated in the table below, the Channel Sub-Block consists of a Channel Block Header, a Channel Information Chunk, and a Channel Content Chunk.

A Word About Compression

All channel data in a PSP file can be stored compressed in the LZ77 format, compressed in the Run Length Encoding (RLE) format, or uncompressed (in a "raw" format). This channel data can include composite image color channels, layer color channels, layer transparency mask channels, layer user mask channels, alpha channels, or selection channels.

The composite image and thumbnail color channels are stored in the compression format indicated in the "Composite image compression type" field of the "Composite Image Attributes Information Chunk," as described in Section 5.4.6 of this document. All other channels are stored in the compression format indicated in the "Compression type" field of the "General Image Attributes Chunk," as described in Section 6.3 of this document.

The LZ77 compression scheme used in the PSP file format is the LZ77 variant used in the PNG file format. There are a couple of important distinctions between the way this LZ77 variant is used in the PNG file format and the way it is used in the PSP file format. First, the PNG file format requires that data be sent through one of several filters before it can be compressed, whereas the PSP file format has no such filter. Second, the PNG file format mandates support for a discontiguous compression stream, whereas the PSP file format allows only a contiguous compression stream.

The Run Length Encoding (RLE) scheme used in the PSP file format is a relatively straightforward RLE variant. RLE is a simple compression method that strives to take advantage of contiguous runs of the same data. In the variant used in the PSP file format, the first byte encountered in the compressed stream is a byte that represents a "run count." While decompressing, if this run count is greater than 128, then 128 should be subtracted from the "run count." This calculated run count (run count – 128) indicates the number of times the next byte in the compressed stream should be repeated in the decompressed stream. If the "run count" read from compressed stream is less than 128, then it indicates the number of following bytes (in the

compressed stream) that should be copied as-is from the compressed stream to the decompressed stream. The compressed stream is comprised of these RLE "packets," each containing a "run count" and data associated with that run count.

In all compression methods used in the PSP format, each channel is a single compression stream, which, when decompressed, represents bitmap data stored from left to right and from top to bottom. Each scanline in the image data is stored on a 4 byte boundary.

All channels that represent something other than color (including layer transparency mask channels, layer user mask channels, alpha channels, and selection channels), when decompressed, are 8-bit greyscale bitmaps.

Channel Block Header:

Type	Length	Name	Description
B_ARRAY	4 bytes	Header identifier	Always "`7E 42 4B 00`" (i.e.,"~BK", followed by a zero byte).
WORD	2 bytes	Block identifier	Always PSP_CHANNEL_BLOCK.
DWORD	4 bytes	Total block length	Length of complete Channel Sub-Block, including Channel Information Chunk and Channel Content Chunk.

Channel Information Chunk:

DWORD	4 bytes	Chunk size	Length of Channel Information Chunk.
DWORD	4 bytes	Compressed channel length	Size of the channel in compressed form (for the purpose of determining the length of the last field in this block, let's call this size j).
DWORD	4 bytes	Uncompressed channel length	Size of the channel in uncompressed form.
WORD	2 bytes	Bitmap type	Type of bitmap for which this channel is intended (one of `PSPDIBType`).
WORD	2 bytes	Channel type	Type of channel (one of `PSPChannelType`).
	unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Channel Content Chunk:

B_ARRAY	j bytes	Channel content	Content of channel. This field contains (possibly compressed) data that defines the channel. Other blocks define whether the channel is compressed and define the size of the bitmap associated with this channel (height and width).

6 Paint Shop Pro (PSP) File Format Description

This section details the PSP file format.

6.1 Constraints

This section describes miscellaneous constraints that writers of the PSP file format must follow.

6.1.1 Main Block Ordering

With regard to the order of the ten main blocks, there are two hard and fast rules. First, the General Image Attributes block must come immediately after the File Header. Second, the Composite Image Bank Block, if present, must come before the Layer Bank Block.

The order of other blocks is not mandated. However, the order in which the main blocks are illustrated in Table 1 of section "4 : Format Overview," is the preferred order. This order allows programs to obtain information quickly about an image document (information such as bit depth, compression type, resolution, etc.) without having to read layer information (such as color bitmap and mask data).

6.2 PSP File Header

Every PSP file begins with the following header, which should be used for initial validation. Any file that does not begin with this header should be considered an invalid PSP file.

As illustrated in the following table, the PSP file header consists of the PSP file signature followed by a version number.

Type	Length	Name	Description
B_ARRAY	32 bytes	PSP file signature	Always "50 61 69 6E 74 20 53 68 6F 70 20 50 72 6F 20 49 6D 61 67 65 20 46 69 6C 65 0A 1A 00 00 00 00" (i.e., the string "Paint Shop Pro Image File\n\x1a", padded with zeroes to 32 bytes).
WORD	2 bytes	PSP file major version number	Currently 6. PSP files produced by Paint Shop Pro 5 have major version number 3. PSP files produced by Paint Shop Pro 6 have major version number 4. PSP files produced by Paint Shop Pro 7 have major version number 5. PSP files produced by Paint Shop Pro 8 have major version number 6. PSP files produced by Paint Shop Pro 9 have major version number 7.
WORD	2 bytes	PSP file minor version	Currently 0.

6.3 The General Image Attributes Block

The General Image Attributes Block, a required block that contains various information about the stored image document, immediately follows the PSP file header. Much of this information is needed to read other parts of the file. (For example, the compression type is required to decode any channel blocks (with the exception of the Composite Image Bank Block channel(s)), and the layer count is required to know how many layers need to be read from the Layers Block).

As illustrated in the following table, the General Image Attributes Block consists of the General Image Attributes Header and the General Image Attributes Chunk.

General Image Attributes Header:

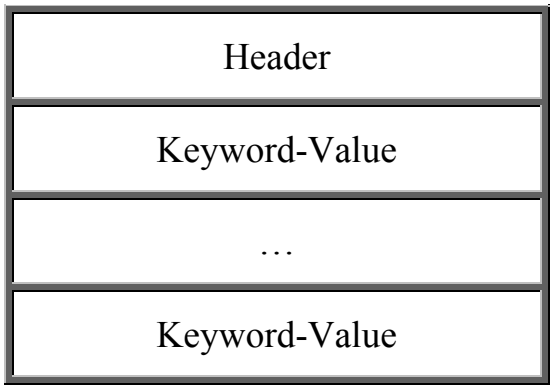
Type	Length	Name	Description
B_ARRAY	4 bytes	Header identifier	Always "7E 42 4B 00" (i.e.,"~BK", followed by a zero byte).
WORD	2 bytes	Block identifier	Always PSP_IMAGE_BLOCK.
DWORD	4 bytes	Total block length	Length of complete General Image Attributes Block.

General Image Attributes Chunk:

DWORD	4 bytes	Chunk size	Length of General Image Attributes Chunk.
LONG	4 bytes	Image width	Width of the image.
LONG	4 bytes	Image height	Height of the image.
DOUBLE	8 bytes	Resolution value	Number of pixels per metric.
BYTE	1 byte	Resolution metric	Metric used for resolution (one of PSP_METRIC).
WORD	2 bytes	Compression type	Type of compression used to compress all image document channels, except those in the composite images, which have their own compression type field (one of PSPCompression). The compression type PSP_COMP_JPEG is not valid here (only used for composite images).
WORD	2 bytes	Bit depth	The bit depth of the color bitmap in each layer of the image document (must be 1, 4, 8, 24, or 48).
WORD	2 byte	Plane count	Number of planes in each layer of the image document (this value must be 1).
DWORD	4 bytes	Color count	Number of colors in each layer of the image document ($2^{\text{Bit depth}}$).
BYTE	1 byte	Greyscale flag	Indicates whether the color bitmap in each layer of image document is a greyscale (0 = not greyscale, 1 = greyscale).
DWORD	4 byte	Total image size	Sum of the sizes of all layer color bitmaps.
LONG	4 bytes	Active layer	Identifies the layer that was active when the image document was saved.
WORD	2 bytes	Layer count	Number of layers in the document.

DWORD	4 bytes	Graphic contents	A series of flags (in PSPGraphicContents) that helps define the image's graphic contents.
	unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

6.4 The Extended Data Block



The Extended Data Block is an optional block that contains miscellaneous information about the image document. This information is organized in the PSP field format, which consists of a series of keyword-value pairs.

Using keyword-value pairs in the Extended Data Block facilitates the storage of new, currently undefined, image document attributes. Applications can be modified quickly to make use of the new keywords, while old applications that do not know about the new keywords will simply ignore them. (Therefore, unknown keywords encountered in the Extended Data Block should be ignored.) If the Extended Data Block is present, it must contain at least one Extended Data Field.

As illustrated in the following table, the Extended Data Block consists of a block header followed by one or more Extended Data Fields (each Extended Data Field consists of a keyword chunk and a value chunk).

Extended Data Block Header:

Type	Length	Name	Description
B_ARRAY	4 bytes	Header identifier	Always "`7E 42 4B 00`" (i.e.,"~BK", followed by a zero byte).
WORD	2 bytes	Block identifier	Always PSP_EXTENDED_DATA_BLOCK.
DWORD	4 bytes	Total block length	Length of following Extended Data Block (includes all following extended data fields – each of which consists of an Extended Data Keyword Chunk and an Extended Data Value Chunk).

Extended Data Keyword Chunk (first extended data field):

B_ARRAY	4 bytes	Field signature	Always "`7E 46 4C 00`" (i.e.,"~FL", followed by a zero byte).
WORD	2 bytes	Field keyword	Keyword identifying type of extended data that is contained in the Extended Data Value Chunk (must

			be one of PSPExtendedDataID).
DWORD	4 bytes	Extended Data Value Chunk size	Length of following Extended Data Value Chunk (lets call this k).

Extended Data Value Chunk (first extended data field):

B_ARRAY	k bytes	Field value	The contents of the Extended Data Value Chunk are dependent upon the keyword extracted from the Extended Data Keyword Chunk (currently there are three defined keywords). Unknown keywords should be ignored:	
			Field keyword:	Field value description:
			PSP_XDATA_TRNS_INDEX	A WORD that identifies the palette index of the transparent color (this field is not valid for image documents whose color bitmaps are non-paletted and should be ignored in this case).
			PSP_XDATA_GRID	This field is 14 bytes and contains the image grid information. The contents are as described in the Grid field value section below.
			PSP_XDATA_GUIDE	This field is 10 bytes and contains information about one image guide (there can be multiple guides). The contents are as described in the Guide field value section below.
			PSP_XDATA_EXIF	This field contains EXIF-specific attribute information. The contents are as described in the EXIF field value section below.
			PSP_XDATA_IPTC	This field contains IPTC-specific attribute information. The contents are as described in the IPTC field value section below.

• • •

Extended Data Keyword Chunk (last extended data field):

B_ARRAY	4 bytes	Field identifier	Always "`7E 46 4C 00`" (i.e.,"~FL", followed by a zero byte).
WORD	2 bytes	Field keyword	Keyword identifying type of extended data that is contained in the Extended Data Value Chunk (must be one of PSPExtendedDataID).
DWORD	4 bytes	Data length	Length of following Extended Data Value Chunk (lets call this k).

Extended Data Value Chunk (last extended data field):

B_ARRAY	k bytes	Field value	The contents of the Extended Data Value Chunk are dependent upon the keyword extracted from the Extended Data Keyword Chunk (currently there are five defined keywords). Unknown keywords should be ignored:	
			Field keyword:	Field value description:
			PSP_XDATA_TRNS_INDEX	A WORD that identifies the palette index of the transparent color (this field is not valid for image documents whose color bitmaps are non-paletted and should be ignored in this case).
			PSP_XDATA_GRID	This field is 14 bytes and contains the image grid information. The contents are as described in the Grid field value section below.
			PSP_XDATA_GUIDE	This field is 10 bytes and contains information about one image guide (there can be multiple guides). The contents are as described in the Guide field value section below.
			PSP_XDATA_EXIF	This field contains EXIF-specific attribute information. The contents are as described in the EXIF field value section below.
			PSP_XDATA_IPTC	This field contains IPTC-specific attribute information. The contents are as described in the IPTC field value section below.

Grid field value contents:

DWORD	4 bytes	Color	Grid color (RGB).
DWORD	4 bytes	Horizontal spacing	Horizontal grid spacing.
DWORD	4 bytes	Vertical spacing	Vertical grid spacing.
WORD	2 bytes	Units	Grid spacing units (must be one of PSPGridUnitsType).

Guide field value contents:

DWORD	4 bytes	Color	Guide color (RGB).
DWORD	4 bytes	Offset	Guide offset (position of guide in document).
WORD	2 bytes	Orientation	Guide orientation (must be one of PSPGuideOrientationType).

EXIF field value contents:

B_ARRAY	6 bytes	EXIF header	Always "45 78 69 66 00 00" (i.e.,"Exif", followed by two zero bytes).
B_ARRAY	8 bytes	TIFF header	TIFF header as defined by TIFF Rev. 6.0. Contains byte-order identifier (WORD), version number (WORD), and offset of the first IFD (DWORD). The first IFD is the following EXIF IFD.
B_ARRAY	k - 14 bytes	EXIF IFD	This field contains a set of tags with EXIF-specific attribute information. The format of this IFD value is defined by the EXIF version 2.2 specification (http://www.exif.org/specifications.html). This field does not contain thumbnail data. The one exception to the EXIF standard is when makernotes are stored this is explained below.

For the EXIF IFD field, the tag that is stored differently from the EXIF specification is the makernote field. It is stored either in straight binary format inside its own IFD, or stored as binary and as TIFF tags decoded from the binary stream. The storage type depends on camera the file originated from and will dictate the 'makernote type'. The 'makernote type', is one of 5 different types that will depend on if the binary makernote can be decoded or not:

Unknown - 0 (the binary makernote stream could not be decoded or recognized)
IFD - 1
IFD prefixed - 2
TIFF header prefixed - 3
TIFF prefixed offset II - 4

Here's the format:

Makernote IFD: 37500 – same ID as in the EXIF specification (current for EXIF 2.2)

Makernote type (see types above)

Depending on the type the rest follows this format

Unknown:
Binary makernote (ID: 1000)

IFD:
Makernote IFD (1001)
Individual makernote tag 1
Individual makernote tag 2
…
Individual makernote tag n
Binary makernote (ID: 1000)

IFD prefixed
Makernote IFD Prefix (ID: 2)
Makernote IFD (1001)
Individual makernote tag 1
Individual makernote tag 2
…
Individual makernote tag n
Binary makernote (ID: 1000)

TIFF Header prefixed – same format as IFD Prefixed
TIFF prefixed offset II – same format as IFD Prefixed

IPTC field value contents:

DWORD	4 bytes	Size	Total number of all IPTC DataSets

DataSet Chunk (first value)

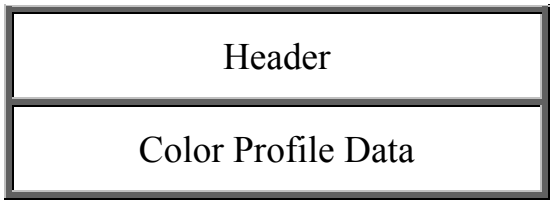
DWORD	4 bytes	Record Number	PSP always uses 2 for "Application Record". May be extended in the future to allow other record types.
DWORD	4 bytes	Dataset ID	Identifier for the data field. See chapter 6 of IPTC specification ("Application record") for valid values.
BYTE	1 byte	DataSet type	Always set to 1 to indicate extended dataset. 0 indicates standard dataset format (not used).
DWORD	4 bytes	Size of data	Indicates size in bytes of data "N"
B_ARRAY	N bytes	DataSet data	An array of bytes containing the data

• • •

DataSet Chunk (last value)

DWORD	4 bytes	Record Number	PSP always uses 2 for "Application Record". May be extended in the future to allow other record types.
DWORD	4 bytes	Dataset ID	Identifier for the data field. See chapter 6 of IPTC specification ("Application record") for valid values.
BYTE	1 byte	DataSet type	Always set to 1 to indicate extended dataset. 0 indicates standard dataset format (not used).
DWORD	4 bytes	Size of data	Indicates size in bytes of data "N"
B_ARRAY	N bytes	DataSet data	An array of bytes containing the data

6.5 The Color Profile Block



The Color Profile Data Block is an optional block that contains ICC color profile data that is used in the Paint Shop Pro color management system when converting between color spaces.

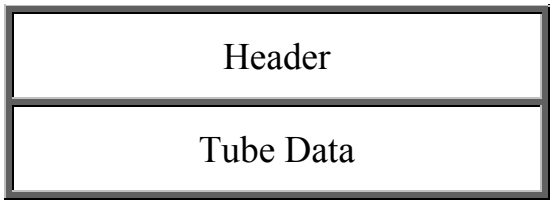
Color Profile Data Block Header:

Type	Length	Name	Description
B_ARRAY	4 bytes	Header identifier	Always "7E 42 4B 00" (i.e.,"~BK", followed by a zero byte).
WORD	2 bytes	Block identifier	Always PSP_COLORPROFILE_BLOCK.
DWORD	4 bytes	Total block length	Length of following complete Color Profile Data Block.

Color Profile Data Chunk:

DWORD	4 bytes	Chunk size	Length of Color Profile Data chunk.
WORD	2 byte	Description Size	Size of the text description of the color profile
B_ARRAY	n bytes	Description	A text description of the color profile
DWORD	4 bytes	Size of color profile	Size of raw color profile data in bytes
B_ARRAY	n bytes	ICC Color Profile	The raw color profile data in ICC format.

6.6 The Tube Data Block



The Tube Data Block is an optional block that contains Paint Shop Pro Picture Tube control data. This block controls how Picture Tube files (which are PSP files) act.

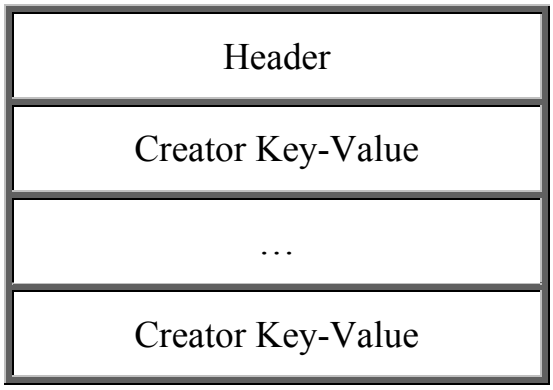
Tube Data Block Header:

Type	Length	Name	Description
B_ARRAY	4 bytes	Header identifier	Always "7E 42 4B 00" (i.e.,"~BK", followed by a zero byte).
WORD	2 bytes	Block identifier	Always PSP_TUBE_BLOCK.
DWORD	4 bytes	Total block length	Length of following complete Tube Data Block.

Tube Data Chunk:

DWORD	4 bytes	Chunk size	Length of Tube Data chunk.
WORD	2 bytes	Block version	Version of following Tube Data Block.
LONG	4 bytes	Tube step size	Tube step size.
LONG	4 byte	Tube column count	Number of columns in tube file.
LONG	4 bytes	Tube row count	Number of rows in tube.
LONG	4 bytes	Tube cell count	Number of cells in tube.
LONG	4 bytes	Tube placement mode	Controls tube placement mode (one of TubePlacementMode).
LONG	4 bytes	Tube selection mode	Controls tube selection mode (one of TubeSelectionMode).
LONG	4 bytes	Tube scale factor	controls size/scale of tube. Valid range is 10-250
	unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

6.7 The Creator Data Block



The Creator Data Block is an optional block that contains miscellaneous information about the creator of the image document. This information is organized in the PSP field format, which consists of a series of keyword-value pairs.

Using keyword-value pairs in the Creator Data Block facilitates the storage of new, currently undefined, creator attributes. Applications can be modified quickly to make use of the new keywords, while old applications that do not know about the new keywords will simply ignore them; (therefore, unknown keywords encountered in the Creator Data Block should be ignored). If the Creator Data Block is present, it must contain at least one creator field.

As illustrated in the following table, the Creator Data Block consists of a block header followed by one or more Creator Data Fields (each Creator Data Field consists of a keyword chunk and a value chunk).

Creator Block Header:

Type	Length	Name	Description
B_ARRAY	4 bytes	Header identifier	Always "7E 42 4B 00" (i.e.,"~BK", followed by a zero byte).
WORD	2 bytes	Block identifier	Always PSP_CREATOR_BLOCK.
DWORD	4 bytes	Total block length	Length of following Creator Block (includes all following creator data fields, each of which consists of a Creator Keyword Chunk and a Creator Value Chunk).

Creator Keyword Chunk (first creator field):

B_ARRAY	4 bytes	Field identifier	Always "7E 46 4C 00" (i.e.,"~FL", followed by a zero byte).
WORD	2 bytes	Field keyword	Keyword identifying the type of data that is contained in the Creator Value Chunk (must be one of PSPCreatorFieldID).
DWORD	4 bytes	Data length	Length of the Creator Value Chunk (lets call this n).

Creator Value Chunk (first creator field):

B_ARRAY	n bytes	Field value	The contents of the field value are dependent upon the field keyword extracted from the previous Creator Keyword Chunk:	
			Field keyword:	Field value description
			PSP_CRTR_FLD_TITLE	Title of image document (in ASCII format).
			PSP_CRTR_FLD_CRT_DATE	DWORD indicating date of image document creation (in MSVC time_t format).
			PSP_CRTR_FLD_MOD_DATE	DWORD indicating date of last image document modification (in MSVC time_t format).
			PSP_CRTR_FLD_ARTIST	Name of artist (in ASCII text).
			PSP_CRTR_FLD_CPYRGHT	Name of copyright holder (in ASCII text).
			PSP_CRTR_FLD_DESC	Description of image document (in ASCII text).
			PSP_CRTR_FLD_APP_ID	DWORD identifying application that created image document (one of PSPCreatorAppID).
			PSP_CRTR_FLD_APP_VER	DWORD identifying version of application that created image document.

• • •

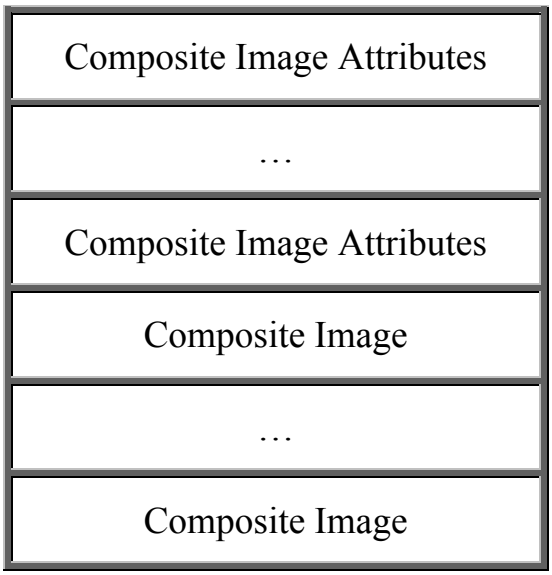
Creator Keyword Chunk (last creator field):

B_ARRAY	4 bytes	Field identifier	Always "7E 46 4C 00" (i.e.,"~FL", followed by a zero byte).
WORD	2 bytes	Field keyword	Keyword identifying the type of data that is contained in the Creator Value Chunk (must be one of PSPCreatorFieldID).
DWORD	4 bytes	Data length	Length of the Creator Value Chunk (lets call this p).

Creator Value Chunk (last creator field):

B_ARRAY	p bytes	Field value	The contents of the field value are dependent upon the field keyword extracted from the previous Creator Keyword Chunk:	
			Field keyword:	Field value description:
			PSP_CRTR_FLD_TITLE	Title of text (in ASCII format).
			PSP_CRTR_FLD_CRT_DATE	DWORD indicating date of image document creation (in MSVC time_t format).
			PSP_CRTR_FLD_MOD_DATE	DWORD indicating date of last image document modification (in MSVC time_t format).
			PSP_CRTR_FLD_ARTIST	Name of artist (in ASCII text).
			PSP_CRTR_FLD_CPYRGHT	Name of copyright holder (in ASCII text).
			PSP_CRTR_FLD_DESC	Description of image document (in ASCII text).
			PSP_CRTR_FLD_APP_ID	DWORD identifying application that created image document (one of PSPCreatorAppID).
			PSP_CRTR_FLD_APP_VER	DWORD identifying version of application that created image document.

6.8 The Composite Image Bank Block



The Composite Image Bank Block is a block that contains one or more bitmaps of the merged document. Typically, the Composite Image Bank Block will contain one miniature bitmap of the merged document (a thumbnail) and one full size bitmap of the merged document. The full size bitmap must have the same bit depth and resolution as the entire image. All PSP files with more than one layer or with one layer with transparency must have a full size composite image. Any additional composite images will generally have different bit depths or different resolutions.

The Composite Image Bank Block is organized such that information about all the composite images is presented first, followed by all the composite images themselves. The block is organized this way so the reader can peruse the sizes and bit depths of all composite images, choose the desired composite image, and then read only the desired composite image (skipping past the other composite images). The composite image information blocks and the composite image blocks are stored in the same order (i.e., the first composite image information block contains information about the first composite image).

As illustrated below, the Composite Image Bank Block consists of a Block Header, a Composite Image Bank Info Chunk, the Composite Image Attributes entries, and the Composite Image entries.

Composite Image Bank Block Header:

Type	Length	Name	Description
B_ARRAY	4 bytes	Header identifier	Always "`7E 42 4B 00`" (i.e.,"~BK", followed by a zero byte).
WORD	2 bytes	Block identifier	Always PSP_COMPOSITE_IMAGE_BANK_BLOCK.
DWORD	4 bytes	Total block length	Length of complete Composite Image Bank Block, including Composite Image Bank Information Chunk, Composite Image Attribute Entries Chunk, and the Composite Image Entries Chunk.

Composite Image Bank Information Chunk:

DWORD	4 bytes	Chunk size	Length of Composite Image Bank Information chunk.

DWORD	4 bytes	Composite image count	Number of composite images in the bank (for the purpose of determining the number of fields in the Composite Image Attributes Entries Chunk, let's call this i).
	Unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

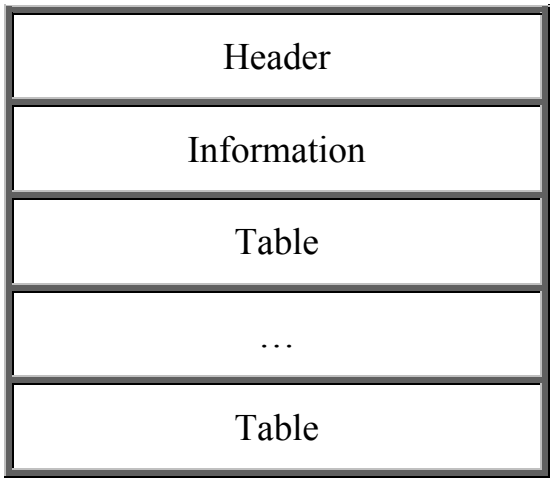
Composite Image Attributes Entries Chunk

Composite Image Attributes Block	See Section 5.4.6.	Entry 1	The composite image attributes of the 1st composite image. Please See Section "5.4.6:Composite Image Attributes Sub-Block" for more information about composite image attributes.
…	…	…	…
Composite Image Attributes Block	See Section 5.4.6.	Entry i	The composite image attributes of the ith composite image. Please See Section "5.4.6:Composite Image Attributes Sub-Block" for more information about composite image attributes.

Composite Image Entries Chunk

Composite Image Block	See Section 5.4.7.	Entry 1	The 1st composite image. Please See Section "5.4.7:Composite Image Sub-Block" for more information about composite images.
…	…	…	…
Composite Image Block	See Section 5.4.7.	Entry i	The ith composite image. Please See Section "5.4.7:Composite Image Sub-Block" for more information about composite images.

6.9 The Table Bank Block



The Table Bank Block is an optional block that contains tables of information associated with the image document. For example, it contains paper and pattern vector tables used by the paper and pattern paint styles in the Paint Style Sub-Block.

As illustrated below, the Table Bank Block consists of a Block Header, a Table Bank Information Chunk, and a separate Table Sub-Block for each table. The number of tables in the Table Bank Block is indicated in the Table Bank Information Chunk.

Table Bank Block Header:

Type	Length	Name	Description
B_ARRAY	4 bytes	Header identifier	Always "7E 42 4B 00" (i.e.,"~BK", followed by a zero byte).
WORD	2 bytes	Block identifier	Always PSP_TABLE_BANK_BLOCK.
DWORD	4 bytes	Total block length	Length of complete Table Bank Block, including Table Bank Information Chunk and all Table Sub-Block(s).

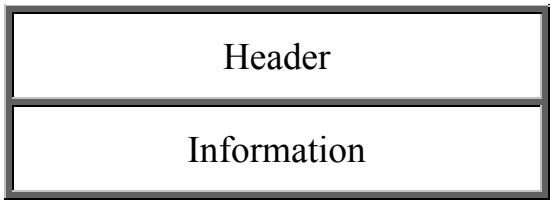
Table Bank Information Chunk:

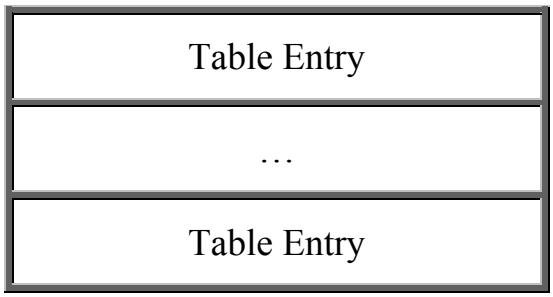
DWORD	4 bytes	Chunk size	Length of Table Bank Information Chunk.
DWORD	4 bytes	Table count	Number of tables in the bank.
	Unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Table Sub-Block(s):

Table Sub-Block	See Section 6.8.1	First table	The first table. Please See Section "6.9.1: Table Sub-Block" for more information about tables.
…	…	…	…
Table Sub-Block	See Section 6.8.1	Last table	The last table. Please See Section "6.9.1: Table Sub-Block" for more information about tables.

6.9.1 Table Sub-Block





The Table Sub-Block defines a table in the Table Bank Block. As illustrated below, the Table Sub-Block consists of the Table Sub-Block Header, the Table Information Chunk, and the Table Entry Sub-Block(s). The content of each Table Entry Sub-Block is based on the table type defined in the Table Information Chunk.

Table Block Header:

Type	Length	Name	Description
B_ARRAY	4 bytes	Header identifier	Always "7E 42 4B 00" (i.e.,"~BK", followed by a zero byte).
WORD	2 bytes	Block identifier	Always PSP_TABLE_BLOCK.
DWORD	4 bytes	Total block length	Length of complete Table Sub-Block, including Table Information Chunk and all Table Entry Sub-Block(s).

Table Information Chunk:

DWORD	4 bytes	Chunk size	Length of Table Information Chunk.
WORD	2 bytes	Table type	Type of table (must be one of PSPTableType).
DWORD	4 bytes	Table size	Number of entries in the table. The content of each entry is based on the table type.
	Unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Table Entry Sub-Block(s):

Table Entry Sub-Block	See Section 6.8.1.1	First table entry	The first table entry. Please See Section "6.9.1.1: Table Entry Sub-Block" for more information about table entries.
…	…	…	…
Table Entry Sub-Block	See Section 6.8.1.1	Last table entry	The last table entry. Please See Section "6.9.1.1: Table Entry Sub-Block" for more information about table entries.

6.9.1.1 Table Entry Sub-Block

The Table Entry Sub-Block contains one table entry in the Table Sub-Block. The contents of this Table Entry Sub-Block are based on the table type defined in the Table Information Chunk.

If the table type is keTTPaperTable, the contents are as described in the section "6.9.1.1.1: Paper Sub-Block."

If the table type is keTTPatternTable, the contents are as described in the section "6.9.1.1.2: Pattern Sub-Block."

6.9.1.1.1 Paper Sub-Block

The Paper Sub-Block defines a table entry in the Table Sub-Block if the table type is keTTPaperTable. As illustrated in the following table, the Paper Sub-Block consists of a Block Header, a Paper Information Chunk, a Paper Channel Information Chunk, and a Paper Channel Sub-Block.

Paper Block Header:

Type	Length	Name	Description
B_ARRAY	4 bytes	Header identifier	Always "`7E 42 4B 00`" (i.e.,"~BK", followed by a zero byte).
WORD	2 bytes	Block identifier	Always PSP_PAPER_BLOCK.
DWORD	4 bytes	Total block length	Length of complete Paper Sub-Block, including Paper Information Chunk, Paper Channel Information Chunk, and Paper Channel Sub-Block.

Paper Information Chunk:

DWORD	4 bytes	Chunk size	Length of Paper Information Chunk.
Variable length string chunk	See Section 5.4.2	Paper name	Paper texture name (not including file path and extension). (Please See Section "5.4.2 – Variable Length String Chunk" for more information about variable string chunks.)
LONG	4 bytes	Paper bitmap width	Specifies the width of the paper bitmap, in pixels.
LONG	4 bytes	Paper bitmap height	Specifies the height of the paper bitmap, in pixels.
	Unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Paper Channel Information Chunk:

DWORD	4 bytes	Chunk size	Length of Paper Channel Information Chunk.
WORD	2 bytes	Bitmap count	Number of paper bitmaps to follow. (Currently always 1 because the paper bitmap is stored as an 8-bit greyscale bitmap.)

WORD	2 bytes	Channel count	Number of channels to follow. (Currently always 1.)
	Unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Paper Channel Sub-Block:

Channel Sub-Block defining the paper bitmap. See Section "5.4.8: Channel Sub-Block and Channel Compression." Note that a paper bitmap is stored as an 8-bit greyscale bitmap.

6.9.1.1.2 Pattern Sub-Block

The Pattern Sub-Block defines a table entry in the Table Sub-Block if the table type is keTTPatternTable. As illustrated in the following table, the Pattern Sub-Block consists of a Block Header, a Pattern Information Chunk, a Pattern Channel Information Chunk, a Pattern Color Palette Sub-Block (if any), and the Pattern Channel Sub-Blocks.

Pattern Block Header:

Type	Length	Name	Description
B_ARRAY	4 bytes	Header identifier	Always "`7E 42 4B 00`" (i.e.,"~BK", followed by a zero byte).
WORD	2 bytes	Block identifier	Always PSP_PATTERN_BLOCK.
DWORD	4 bytes	Total block length	Length of complete Pattern Sub-Block, including Pattern Information Chunk, Pattern Channel Information Chunk, Pattern Color Palette Sub-Block (if any), and all Pattern Channel Sub-Blocks.

Pattern Information Chunk:

DWORD	4 bytes	Chunk size	Length of Pattern Information Chunk.
Variable length string chunk	See Section 5.4.2	Pattern name	Pattern name (not including file path and extension). (Please See Section "5.4.2 – Variable Length String Chunk" for more information about variable string chunks.)
LONG	4 bytes	Pattern width	Specifies the width of the pattern bitmap, in pixels.
LONG	4 bytes	Pattern height	Specifies the height of the pattern bitmap, in pixels.
WORD	2 bytes	Pattern bit depth	Number of bits used to represent each color pixel of the pattern bitmap (must be 1, 4, 8, or 24).
BYTE	1 byte	Pattern greyscale flag	Indicates whether the pattern bitmap is a greyscale bitmap (0 = not greyscale, 1 = greyscale).
	Unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Pattern Channel Information Chunk:

DWORD	4 bytes	Chunk size	Length of Pattern Channel Information Chunk.
WORD	2 bytes	Bitmap count	Number of bitmaps to follow (1 if the pattern has no transparency mask; 2 if the pattern has a transparency mask).
WORD	2 bytes	Channel count	Number of channels to follow.
	Unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

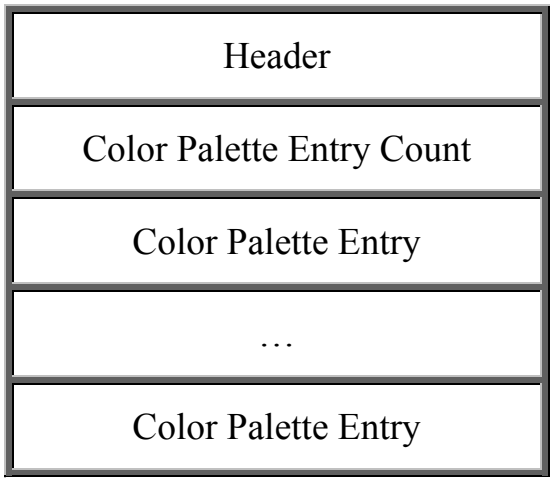
Pattern Color Palette Sub-Block:

Color Palette Sub-Block containing the pattern's color palette (this Sub-Block is not present in 24-bit or 48-bit pattern bitmaps). See Section "5.4.3 : Color Palette Sub-Block."

Pattern Channel Sub-Blocks:

All the pattern channels. Possible channels include: 1) One or three Channel Sub-Blocks defining the pattern bitmap. If the pattern bitmap is a paletted bitmap, there is one channel; if the pattern bitmap is a 24 or 48 bit bitmap, there are three channels. 2) One channel defining the pattern transparency mask. The transparency mask is stored as an 8-bit greyscale bitmap. See Section "5.4.8: Channel Sub-Block and Channel Compression."

6.10 The Color Palette Block

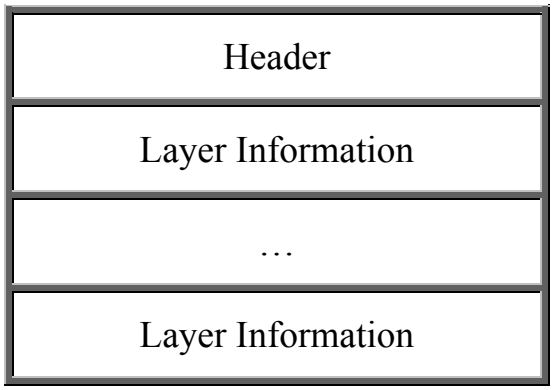


The Color Palette Block contains a global color palette to be used with the color bitmaps in all layers of the document. This block is required for image documents that contain paletted bitmaps and should not be written for image documents that contain 24-bit or 48-bit bitmaps (the block can be ignored in the latter cases).

As illustrated in the following table, the Color Palette Block consists of a Color Palette Sub-Block.

A Color Palette Block containing the color palette for color bitmaps in all of the image document's layers. This block should not be present in image documents whose color bitmaps are 24-bit or 48-bit bitmaps, and it can be ignored in this case. See Section "5.4.3 : Color Palette Sub-Block."

6.11 The Layer Bank Block



The Layer Bank Block is a required block that contains data describing all the layers in the image document.

As illustrated in the following table, the Layer Bank Block consists of a Block Header and a separate Layer Sub-Block for each layer in the image document (the number of layers in the document is indicated in the General Image Attributes Block). Each Layer Sub-Block consists of a Layer Block Header, a Layer Information Chunk, an optional Layer Extension Sub-Block, and one or more Channel Sub-Blocks.

Layer Bank Block Header:

Type	Length	Name	Description
B_ARRAY	4 bytes	Header identifier	Always "`7E 42 4B 00`" (i.e.,"~BK", followed by a zero byte).
WORD	2 bytes	Block identifier	Always PSP_LAYER_START_BLOCK.
DWORD	4 bytes	Total block length	Length of complete Layers Block, including all Layer Sub-Blocks.

Layer Sub-Block Header (first layer):

Type	Length	Name	Description
B_ARRAY	4 bytes	Header identifier	Always "`7E 42 4B 00`" (i.e.,"~BK", followed by a zero byte).
WORD	2 bytes	Block identifier	Always PSP_LAYER_BLOCK.
DWORD	4 bytes	Total block length	Length of complete Layer Block, including Layer Information Chunk, Layer Extension Sub-Block, Layer Bitmap Chunk, and all Layer Channel Sub-Blocks.

Layer Information Chunk (first layer):

DWORD	4 bytes	Chunk size	Length of Layer Information Chunk.
Variable length string chunk	See Section 5.4.2.	Layer name	Name of layer (please See Section "5.4.2 – Variable Length String Chunk" for more information about variable string chunks).
BYTE	1 byte	Layer type	Type of layer (must be one of PSPLayerType).
RECT	16 bytes	Image rectangle	Rectangle defining image border.
RECT	16 bytes	Saved image rectangle	Rectangle within image rectangle that contains "significant" data (only the contents of this rectangle are saved to the file).
BYTE	1 byte	Layer opacity	Overall layer opacity.
BYTE	1 byte	Blending mode	Mode to use when blending layer (one of PSPBlendModes).
BYTE	1 byte	Layer flags	A series of flags that help define the layer's attributes. (PSPLayerProperties values, bitwise-ored together).
BYTE	1 byte	Transparency protected flag	TRUE if transparency is protected.
BYTE	1 byte	Link group identifier	Identifies group to which this layer belongs.
RECT	16 bytes	Mask rectangle	Rectangle defining user mask border.
RECT	16 bytes	Saved mask rectangle	Rectangle within mask rectangle that contains "significant" data (only the contents of this rectangle are saved to the file).
BYTE	1 byte	Mask linked	TRUE if mask linked to layer (i.e., mask moves relative to layer), FALSE otherwise.
BYTE	1 byte	Mask disabled	TRUE if mask is disabled, FALSE otherwise.
BYTE	1 byte	Invert mask on blend	TRUE if mask should be inverted when the layer is merged, FALSE otherwise.
WORD	2 bytes	Blend range count	Number of valid source-destination field pairs to follow (note, there are currently always 5 such pairs, but they are not necessarily all valid).
B_ARRAY	4 bytes	Source blend range #1	First source blend range value.
B_ARRAY	4 bytes	Destination blend range #1	First destination blend range value.
B_ARRAY	4 bytes	Source blend range #2	Second source blend range value.
B_ARRAY	4 bytes	Destination blend range #2	Second destination blend range value.
B_ARRAY	4 bytes	Source blend range #3	Third source blend range value.
B_ARRAY	4 bytes	Destination blend range #3	Third destination blend range value.

B_ARRAY	4 bytes	Source blend range #4	Fourth source blend range value.
B_ARRAY	4 bytes	Destination blend range #4	Fourth destination blend range value.
B_ARRAY	4 bytes	Source blend range #5	Fifth source blend range value.
B_ARRAY	4 bytes	Destination blend range #5	Fifth destination blend range value.
BYTE	1 byte	Use highlight color	TRUE if use highlight color in layer palette.
DWORD	4 bytes	Highlight color	Highlight color in layer palette (RGB).
	unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Layer Extension Sub-Block (first layer):

The Layer Extension Sub-Block contains data that extends attributes of the current layer. This sub-block must be present for adjustment, group, mask, and vector layers. The contents of the Layer Extension Sub-Block are based on the layer type.

If the layer type is keGLTAdjustment, the contents are as described in the Adjustment Layer Sub-Block (See Section "6.11.1: Adjustment Layer Sub-Block").

If the layer type is keGLTVector, the contents are as described in the Vector Layer Sub-Block (See Section "6.11.2: Vector Layer Sub-Block").

If the layer type is keGLTGroup, the contents are as described in the Group Layer Sub-Block (See Section "6.11.3: Group Layer Sub-Block").

If the layer type is keGLTMask, the contents are as described in the Mask Layer Sub-Block (See Section "6.11.4: Mask Layer Sub-Block").

If the layer type is keGLTArtMedia, the contents are as described in the Art Media Layer Sub-Block (See Section "6.10.5: Art Media Layer Sub-Block").

For any other layer types, the Layer Extension Sub-Block must not be present.

Layer Bitmap Chunk (first layer):

DWORD	4 bytes	Chunk size	Length of Layer Bitmap Chunk.
WORD	2 bytes	Count of bitmaps	Number of bitmaps to follow.
WORD	2 bytes	Channel count	Number of channels to follow.
	unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Layer Channel Sub-Blocks (first layer):

Contains all the layer's channels. Possible channels include 1) one or three Channel Sub-Blocks defining the layer's color bitmap, 2) one channel defining the layer's transparency mask, and 3) one channel defining the layer's user mask or the adjustment layer bitmap. Note that the type of each channel can be obtained from the Channel Sub-Block (See Section "5.4.8: Channel Sub-Block and Channel Compression"), and the number of channels is specified in the previous Layer Information Chunk. Note also that the layer's transparency mask, the layer's user mask, and the adjustment layer bitmap are stored as 8-bit greyscale bitmaps.

• • •

Layer Sub-Block Header (last layer):

Type	Length	Name	Description
B_ARRAY	4 bytes	Header identifier	Always "7E 42 4B 00" (i.e.,"~BK", followed by a zero byte).
WORD	2 bytes	Block identifier	Always PSP_LAYER_BLOCK.
DWORD	4 bytes	Total block length	Length of complete Layers Block, including Layer Information Chunk, Layer Extension Sub-Block, Layer Bitmap Chunk, and all Layer Channel Sub-Blocks.

Layer Information Chunk (last layer):

DWORD	4 bytes	Chunk size	Size of Layer Information Chunk.
Variable length string chunk	See Section 5.4.2.	Layer name	Name of layer (please See Section "5.4.2 – Variable Length String Chunk" for more information about variable string chunks).
BYTE	1 byte	Layer type	Type of layer (must be one of PSPLayerType).
RECT	16 bytes	Image rectangle	Rectangle defining image border.
RECT	16 bytes	Saved image rectangle	Rectangle within image rectangle that contains "significant" data. (Only the contents of this rectangle are saved to the file.)
BYTE	1 byte	Layer opacity	Overall layer opacity.
BYTE	1 byte	Blending mode	Mode to use when blending layer.
BYTE	1 byte	Layer flags	PSPLayerProperties values, bitwise-ored together.
BYTE	1 byte	Transparency protected flag	TRUE if transparency is protected.
BYTE	1 byte	Link group identifier	Identifies group to which this layer belongs.

RECT	16 bytes	Mask rectangle	Rectangle defining user mask border.
RECT	16 bytes	Saved mask rectangle	Rectangle within mask rectangle that contains "significant" data. (Only the contents of this rectangle are saved to the file.)
BYTE	1 byte	Mask linked	TRUE if mask linked to layer (i.e., mask moves relative to layer), FALSE otherwise.
BYTE	1 byte	Mask disabled	TRUE if mask is disabled, FALSE otherwise.
BYTE	1 byte	Invert mask on blend	TRUE if mask should be inverted when the layer is merged.
WORD	2 bytes	Blend range count	Number of valid source-destination field pairs to follow. (Note, there are currently always 5 such pairs, but they are not necessarily all valid.)
B_ARRAY	4 bytes	Source blend range #1	First source blend range value.
B_ARRAY	4 bytes	Destination blend range #1	First destination blend range value.
B_ARRAY	4 bytes	Source blend range #2	Second source blend range value.
B_ARRAY	4 bytes	Destination blend range #2	Second destination blend range value.
B_ARRAY	4 bytes	Source blend range #3	Third source blend range value.
B_ARRAY	4 bytes	Destination blend range #3	Third destination blend range value.
B_ARRAY	4 bytes	Source blend range #4	Fourth source blend range value.
B_ARRAY	4 bytes	Destination blend range #4	Fourth destination blend range value.
B_ARRAY	4 bytes	Source blend range #5	Fifth source blend range value.
B_ARRAY	4 bytes	Destination blend range #5	Fifth destination blend range value.
BYTE	1 byte	Use highlight color	TRUE if use highlight color in layer palette.
DWORD	4 bytes	Highlight color	Highlight color in layer palette (RGB).
	Unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Layer Extension Sub-Block (last layer):

The Layer Extension Sub-Block contains data that extends attributes of the current layer. This sub-block must be present for adjustment, group, mask, and vector layers. The contents of the Layer Extension Sub-Block are based on the layer type. If the layer type is keGLTAdjustment, the contents are as described in the Adjustment Layer Sub-Block (See Section "6.11.1: Adjustment Layer Sub-Block"). If the layer type is keGLTVector, the contents are as described in the Vector Layer Sub-Block (See Section "6.11.2: Vector Layer Sub-Block").	

If the layer type is keGLTGroup, the contents are as described in the Group Layer Sub-Block (See Section "6.11.3: Group Layer Sub-Block"). If the layer type is keGLTMask, the contents are as described in the Mask Layer Sub-Block (See Section "6.11.4: Mask Layer Sub-Block"). For any other layer types, the Layer Extension Sub-Block must not be present.	

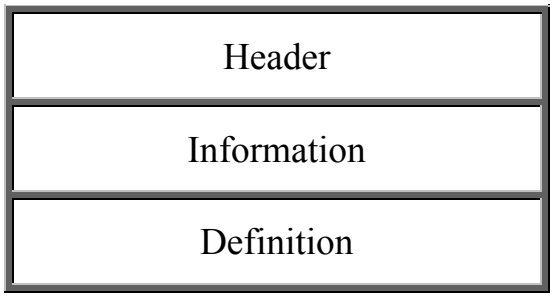
Layer Bitmap Chunk (last layer):

DWORD	4 bytes	Chunk size	Length of Layer Bitmap Chunk.
WORD	2 bytes	Count of bitmaps	Number of bitmaps to follow.
WORD	2 bytes	Channel count	Number of channels to follow.
	unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Layer Channel Sub-Blocks (last layer):

Contains all the layer's channels. Possible channels include 1) one or three Channel Sub-Blocks defining the layer's color bitmap, 2) one channel defining the layer's transparency mask, and 3) one channel defining the layer's user mask or the adjustment layer bitmap. Note that the type of each channel can be obtained from the Channel Sub-Block (See Section "5.4.8: Channel Sub-Block and Channel Compression"), and the number of channels is specified in the previous Layer Information Chunk. Note also that the layer's transparency mask, the layer's user mask, and the adjustment layer bitmap are stored as 8-bit greyscale bitmaps.

6.11.1 Adjustment Layer Sub-Block



The Adjustment Layer Sub-Block contains layer extension data required to define an adjustment layer.

As illustrated below, the Adjustment Layer Sub-Block consists of the Adjustment Layer Block Header, the Adjustment Layer Information Chunk, and the Adjustment Layer Definition Chunk.

Adjustment Layer Block Header:

Type	Length	Name	Description
B_ARRAY	4 bytes	Header identifier	Always "`7E 42 4B 00`" (i.e.,"~BK", followed by a zero byte).
WORD	2 bytes	Block identifier	Always PSP_ADJUSTMENT_EXTENSION_BLOCK.
DWORD	4 bytes	Total block length	Length of complete Adjustment Layer Sub-Block, including Adjustment Layer Information Chunk and Adjustment Layer Definition Chunk.

Adjustment Layer Information Chunk:

DWORD	4 bytes	Chunk size	Length of Adjustment Layer Information Chunk.
WORD	2 bytes	Adjustment layer type	Type of adjustment layer (must be one of PSPAdjustmentLayerType).
	Unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Adjustment Layer Definition Chunk:

Adjustment Layer Definition. The contents of this chunk are based on the previous field (adjustment layer type). If the adjustment layer type is keAdjLevel, the contents are as described in the Level Adjustment Layer Definition Chunk (below). If the adjustment layer type is keAdjCurve, the contents are as described in the Curve Adjustment Layer Definition Chunk (below). If the adjustment layer type is keAdjBrightContrast, the contents are as described in the Brightness / Contrast Adjustment Layer Definition Chunk (below). If the adjustment layer type is keAdjColorBal, the contents are as described in the Color Balance Adjustment Layer Definition Chunk (below). If the adjustment layer type is keAdjHSL, the contents are as described in the HSL Adjustment Layer Definition Chunk (below). If the adjustment layer type is keAdjChannelMixer, the contents are as described in the Channel Mixer Adjustment Layer Definition Chunk (below). If the adjustment layer type is keAdjInvert, the contents are as described in the Invert Adjustment Layer Definition Chunk (below). If the adjustment layer type is keAdjThreshold, the contents are as described in the Threshold Adjustment Layer Definition Chunk (below). If the adjustment layer type is keAdjPoster, the contents are as described in the Posterize Adjustment Layer Definition Chunk (below).

Level Adjustment Layer Definition Chunk:

DWORD	4 bytes	Chunk size	Length of Level Adjustment Layer Definition Chunk.
DOUBLE	8 bytes	Gamma value #1	First gamma value for master, red, green, and blue.
DOUBLE	8 bytes	Gamma value #2	Second gamma value for master, red, green, and blue.
DOUBLE	8 bytes	Gamma value #3	Third gamma value for master, red, green, and blue.
DOUBLE	8 bytes	Gamma value #4	Fourth gamma value for master, red, green, and blue.
LONG	4 bytes	Ceiling input value #1	First ceiling input value for master, red, green, and blue.
LONG	4 bytes	Ceiling input value #2	Second ceiling input value for master, red, green, and blue.
LONG	4 bytes	Ceiling input value #3	Third ceiling input value for master, red, green, and blue.
LONG	4 bytes	Ceiling input value #4	Fourth ceiling input value for master, red, green, and blue.

LONG	4 bytes	Floor input value #1	First floor input value for master, red, green, and blue.
LONG	4 bytes	Floor input value #2	Second floor input value for master, red, green, and blue.
LONG	4 bytes	Floor input value #3	Third floor input value for master, red, green, and blue.
LONG	4 bytes	Floor input value #4	Fourth floor input value for master, red, green, and blue.
LONG	4 bytes	Ceiling output value #1	First ceiling output value for master, red, green, and blue.
LONG	4 bytes	Ceiling output value #2	Second ceiling output value for master, red, green, and blue.
LONG	4 bytes	Ceiling output value #3	Third ceiling output value for master, red, green, and blue.
LONG	4 bytes	Ceiling output value #4	Fourth ceiling output value for master, red, green, and blue.
LONG	4 bytes	Floor output value #1	First floor output value for master, red, green, and blue.
LONG	4 bytes	Floor output value #2	Second floor output value for master, red, green, and blue.
LONG	4 bytes	Floor output value #3	Third floor output value for master, red, green, and blue.
LONG	4 bytes	Floor output value #4	Fourth floor output value for master, red, green, and blue.
	unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Curve Adjustment Layer Definition Chunk:

The curve adjustment layer definition consists of four Curve Adjustment Layer Definitions Chunks, one for each of the four curve channels RGB, Red, Green, and Blue.

Curve Adjustment Layer Definition Chunk (RGB Channel):

DWORD	4 bytes	Chunk size	Length of Curve Adjustment Layer Definition Chunk.
BYTE	1 byte	Freehand flag	TRUE if curve type is freehand, FALSE if curve type is smooth.
WORD	2 bytes	Point count	Number of points used in the curve (for a smooth curve).
BYTE	1 byte	Channel input value #1	First channel input value in the set of curve points (for a smooth curve). The full set of curve points contains 18 points.
BYTE	1 byte	Channel output value #1	First channel output value in the set of curve points (for a smooth curve). The full set of curve points contains 18 points.
…	…	…	…
BYTE	1 byte	Channel input value #18	Eighteenth channel input value in the set of curve points (for a smooth curve). The full set of curve points contains 18 points.

BYTE	1 byte	Channel output value #18	Eighteenth channel output value in the set of curve points (for a smooth curve). The full set of curve points contains 18 points.
B_ARRAY	256 bytes	Curve lookup table	Lookup table for all values in the channel (for a freehand curve).
	Unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

• • •

Curve Adjustment Layer Definition Chunk (Blue Channel):

DWORD	4 bytes	Chunk size	Length of Curve Adjustment Layer Definition Chunk.
BYTE	1 byte	Freehand flag	TRUE if curve type is freehand, FALSE if curve type is smooth.
WORD	2 bytes	Point count	Number of points used in the curve (for a smooth curve).
BYTE	1 byte	Channel input value #1	First channel input value in the set of curve points (for a smooth curve). The full set of curve points contains 18 points.
BYTE	1 byte	Channel output value #1	First channel output value in the set of curve points (for a smooth curve). The full set of curve points contains 18 points.
…	…	…	…
BYTE	1 byte	Channel input value #18	Eighteenth channel input value in the set of curve points (for a smooth curve). The full set of curve points contains 18 points.
BYTE	1 byte	Channel output value #18	Eighteenth channel output value in the set of curve points (for a smooth curve). The full set of curve points contains 18 points.
B_ARRAY	256 bytes	Curve lookup table	Lookup table for all values in the channel (for a freehand curve).
	Unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Brightness/Contrast Adjustment Layer Definition Chunk:

DWORD	4 bytes	Chunk size	Length of Brightness / Contrast Adjustment Layer Definition Chunk.
LONG	4 bytes	Brightness value	Brightness adjustment value.
LONG	4 bytes	Contrast value	Contrast adjustment value.
	unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Color Balance Adjustment Layer Definition Chunk:

DWORD	4 bytes	Chunk size	Length of Color Balance Adjustment Layer Definition Chunk.
BYTE	1 byte	Preserve Luminance flag	TRUE if luminosity is preserved, FALSE otherwise.
LONG	4 bytes	Highlight value #1	Highlight value for the channel range Cyan to Red.
LONG	4 bytes	Highlight value #2	Highlight value for the channel range Magenta to Green.
LONG	4 bytes	Highlight value #3	Highlight value for the channel range Yellow to Blue.
LONG	4 bytes	Midtone value #1	Midtone value for the channel range Cyan to Red.
LONG	4 bytes	Midtone value #2	Midtone value for the channel range Magenta to Green.
LONG	4 bytes	Midtone value #3	Midtone value for the channel range Yellow to Blue.
LONG	4 bytes	Shadow value #1	Shadow value for the channel range Cyan to Red.
LONG	4 bytes	Shadow value #2	Shadow value for the channel range Magenta to Green.
LONG	4 bytes	Shadow value #3	Shadow value for the channel range Yellow to Blue.
	Unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

HSL Adjustment Layer Definition Chunk:

DWORD	4 bytes	Chunk size	Length of HSL Adjustment Layer Definition Chunk.
BYTE	1 byte	Colorize flag	TRUE if colorize is on, FALSE otherwise.
LONG	4 bytes	Master value #1	First master value (Hue).
LONG	4 bytes	Master value #2	Second master value (Saturation).
LONG	4 bytes	Master value #3	Third master value (Luminance).
LONG	4 bytes	Master colorize value #1	First master value when the colorize option is on (Hue).
LONG	4 bytes	Master colorize value #2	Second master value when the colorize option is on (Saturation).
LONG	4 bytes	Master colorize value #3	Third master value when the colorize option is on (Luminance).
LONG	4 bytes	Red value #1	First red value (Hue).
LONG	4 bytes	Red value #2	Second red value (Saturation).

LONG	4 bytes	Red value #3	Third red value (Luminance).
LONG	4 bytes	Red value #4	Fourth red value. Degree range that defines the red floor-lower range.
LONG	4 bytes	Red value #5	Fifth red value. Degree range that defines the red floor-upper range.
LONG	4 bytes	Red value #6	Sixth red value. Degree range that defines the red ceiling-lower range.
LONG	4 bytes	Red value #7	Seventh red value. Degree range that defines the red ceiling-upper range.
LONG	4 bytes	Yellow value #1	First yellow value (Hue).
LONG	4 bytes	Yellow value #2	Second yellow value (Saturation).
LONG	4 bytes	Yellow value #3	Third yellow value (Luminance).
LONG	4 bytes	Yellow value #4	Fourth yellow value. Degree range that defines the yellow floor-lower range.
LONG	4 bytes	Yellow value #5	Fifth yellow value. Degree range that defines the yellow floor-upper range.
LONG	4 bytes	Yellow value #6	Sixth yellow value. Degree range that defines the yellow ceiling-lower range.
LONG	4 bytes	Yellow value #7	Seventh yellow value. Degree range that defines the yellow ceiling-upper range.
LONG	4 bytes	Green value #1	First green value (Hue).
LONG	4 bytes	Green value #2	Second green value (Saturation).
LONG	4 bytes	Green value #3	Third green value (Luminance).
LONG	4 bytes	Green value #4	Fourth green value. Degree range that defines the green floor-lower range.
LONG	4 bytes	Green value #5	Fifth green value. Degree range that defines the green floor-upper range.
LONG	4 bytes	Green value #6	Sixth green value. Degree range that defines the green ceiling-lower range.
LONG	4 bytes	Green value #7	Seventh green value. Degree range that defines the green ceiling-upper range.
LONG	4 bytes	Cyan value #1	First cyan value (Hue).
LONG	4 bytes	Cyan value #2	Second cyan value (Saturation).
LONG	4 bytes	Cyan value #3	Third cyan value (Luminance).
LONG	4 bytes	Cyan value #4	Fourth cyan value. Degree range that defines the cyan floor-lower range.
LONG	4 bytes	Cyan value #5	Fifth cyan value. Degree range that defines the cyan floor-upper range.
LONG	4 bytes	Cyan value #6	Sixth cyan value. Degree range that defines the cyan ceiling-lower range.
LONG	4 bytes	Cyan value #7	Seventh cyan value. Degree range that defines the cyan ceiling-upper range.

LONG	4 bytes	Blue value #1	First blue value (Hue).
LONG	4 bytes	Blue value #2	Second blue value (Saturation).
LONG	4 bytes	Blue value #3	Third blue value (Luminance).
LONG	4 bytes	Blue value #4	Fourth blue value. Degree range that defines the blue floor-lower range.
LONG	4 bytes	Blue value #5	Fifth blue value. Degree range that defines the blue floor-upper range.
LONG	4 bytes	Blue value #6	Sixth blue value. Degree range that defines the blue ceiling-lower range.
LONG	4 bytes	Blue value #7	Seventh blue value. Degree range that defines the blue ceiling-upper range.
LONG	4 bytes	Magenta value #1	First magenta value (Hue).
LONG	4 bytes	Magenta value #2	Second magenta value (Saturation).
LONG	4 bytes	Magenta value #3	Third magenta value (Luminance).
LONG	4 bytes	Magenta value #4	Fourth magenta value. Degree range that defines the magenta floor-lower range.
LONG	4 bytes	Magenta value #5	Fifth magenta value. Degree range that defines the magenta floor-upper range.
LONG	4 bytes	Magenta value #6	Sixth magenta value. Degree range that defines the magenta ceiling-lower range.
LONG	4 bytes	Magenta value #7	Seventh magenta value. Degree range that defines the magenta ceiling-upper range.
	Unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Channel Mixer Adjustment Layer Definition Chunk:

DWORD	4 bytes	Chunk size	Length of Channel Mixer Adjustment Layer Definition Chunk.
BYTE	1 byte	Monochrome flag	TRUE if monochrome, FALSE otherwise.
LONG	4 bytes	Blue channel value #1	First value for the blue output channel (Red).
LONG	4 bytes	Blue channel value #2	Second value for the blue output channel (Green).
LONG	4 bytes	Blue channel value #3	Third value for the blue output channel (Blue).
LONG	4 bytes	Blue channel value #4	Fourth value for the blue output channel (Constant).
LONG	4 bytes	Green channel value #1	First value for the green output channel (Red).
LONG	4 bytes	Green channel value #2	Second value for the green output channel (Green).
LONG	4 bytes	Green channel value #3	Third value for the green output channel (Blue).

LONG	4 bytes	Green channel value #4	Fourth value for the green output channel (Constant).
LONG	4 bytes	Red channel value #1	First value for the red output channel (Red).
LONG	4 bytes	Red channel value #2	Second value for the red output channel (Green).
LONG	4 bytes	Red channel value #3	Third value for the red output channel (Blue).
LONG	4 bytes	Red channel value #4	Fourth value for the red output channel (Constant).
	unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Invert Adjustment Layer Definition Chunk:

DWORD	4 bytes	Chunk size	Length of Invert Adjustment Layer Definition Chunk.
	unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

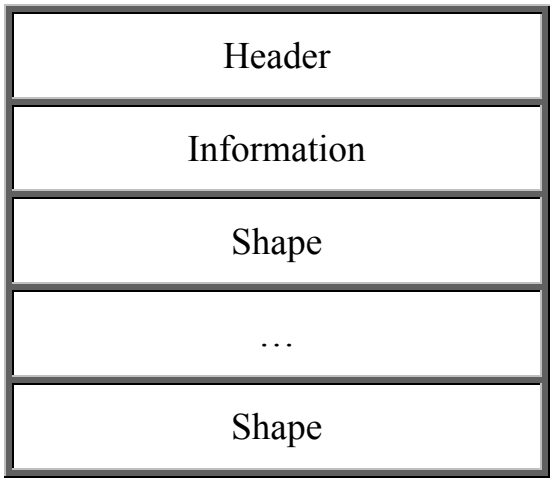
Threshold Adjustment Layer Definition Chunk:

DWORD	4 bytes	Chunk size	Length of Threshold Adjustment Layer Definition Chunk.
LONG	4 bytes	Threshold value	Threshold value.
	unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Posterize Adjustment Layer Definition Chunk:

DWORD	4 bytes	Chunk size	Length of Posterize Adjustment Layer Definition Chunk.
LONG	4 bytes	Posterize value	Posterize value.
	unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

6.11.2 Vector Layer Sub-Block



The Vector Layer Sub-Block contains layer extension data required to define a vector layer.

As illustrated below, the Vector Layer Sub-Block consists of a Vector Layer Block Header, a Vector Layer Information Chunk, and a separate Vector Shape Sub-Block for each vector shape. (The number of vector shapes for the vector layer is indicated in the Vector Layer Information Chunk.)

Vector Layer Block Header:

Type	Length	Name	Description
B_ARRAY	4 bytes	Header identifier	Always "`7E 42 4B 00`" (i.e.,"~BK", followed by a zero byte).
WORD	2 bytes	Block identifier	Always PSP_VECTOR_EXTENSION_BLOCK.
DWORD	4 bytes	Total block length	Length of complete Vector Layer Sub-Block, including the Vector Layer Information Chunk and the Vector Shape Sub-Block(s).

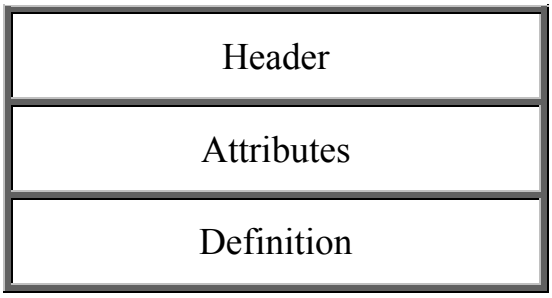
Vector Layer Information Chunk:

DWORD	4 bytes	Chunk size	Length of Vector Layer Information Chunk.
DWORD	4 bytes	Shape count	Number of vector shapes (i.e., number of following Vector Shape Sub-Block(s)).
	Unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Vector Shape Sub-Block(s):

Vector Shape Sub-Block	See Section 6.9.2.1	First Vector Shape	The first vector shape. Please See Section "6.11.2.1: Vector Shape Sub-Block" for more information about vector shapes.
…	…	…	…
Vector Shape Sub-Block	See Section 6.9.2.1	Last Vector Shape	The last vector shape. Please See Section "6.11.2.1: Vector Shape Sub-Block" for more information about vector shapes.

6.11.2.1 **Vector Shape Sub-Block**



The Vector Shape Sub-Block is used to define one vector shape. As illustrated below, the Vector Shape Sub-Block consists of the Vector Shape Block Header, the Vector Shape Attributes Chunk, and the Vector Shape Definition. The Vector Shape Attributes Chunk contains shape attributes shared by all shape types. The contents of the Vector Shape Definition are based on the shape type defined in the Vector Shape Attributes Chunk.

Vector Shape Block Header:

Type	Length	Name	Description
B_ARRAY	4 bytes	Header identifier	Always "`7E 42 4B 00`" (i.e.,"~BK", followed by a zero byte).
WORD	2 bytes	Block identifier	Always PSP_SHAPE_BLOCK.
DWORD	4 bytes	Total block length	Length of complete Vector Shape Sub-Block, including Vector Shape Attributes Chunk and Vector Shape Definition.

Vector Shape Attributes Chunk:

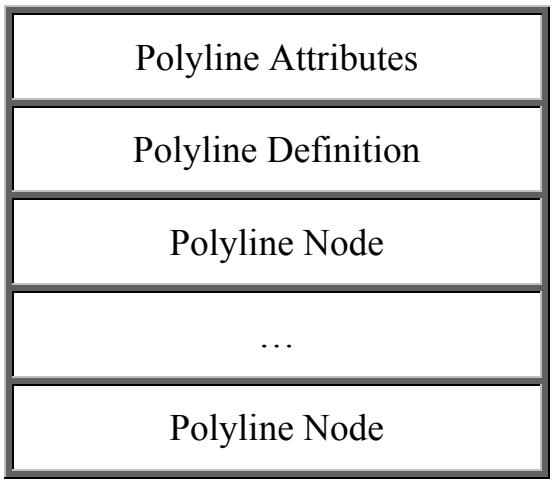
DWORD	4 bytes	Chunk size	Length of Vector Shape Attributes Chunk.
Variable length string chunk	See Section 5.4.2	Shape name	Name of shape. (Please See Section “5.4.2 – Variable Length String Chunk” for more information about variable string chunks.)

WORD	2 bytes	Shape type	Type of vector shape (must be one of PSPVectorShapeType).
DWORD	4 bytes	Shape property flags	A series of flags (in PSPShapeProperties) that defines the shape's properties.
DWORD	4 bytes	Shape identifier	Identifies shape. This identifier must be unique within the layer. This identifier must be 1 based (i.e., 0 is not a valid shape identifier).
DWORD	4 bytes	Link shape identifier	Identifies a shape that is linked to this shape (for text on path). If this value is 0 (an invalid link shape identifier), the shape has no link shape.
	Unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Vector Shape Definition:

The contents of the Vector Shape Definition are based on the shape type (defined in the previous Vector Shape Attributes Chunk). If the shape type is keVSTPolyline, keVSTEllipse, or keVSTPolygon, the contents are as described in the Polyline Vector Shape Definition. If the shape type is keVSTText, the contents are as described in the Text Vector Shape Definition. If the shape type is keVSTGroup, the contents are as described in the Group Vector Shape Definition.

Polyline Vector Shape Definition:



The Polyline Vector Shape Definition consists of the chunks required to define one polyline shape. As illustrated below, the Polyline Vector Shape Definition consists of the Polyline Shape Attributes Chunk, the Polyline Shape Definition Chunk, and the Polyline Node Chunk(s).

Polyline Shape Attributes Chunk:

DWORD	4 bytes	Chunk size	Length of Polyline Shape Attributes Chunk. This size does not include the line paint style, fill

			paint style, and styled line sub blocks.
BYTE	1 byte	Stroked flag	TRUE if stroked, FALSE otherwise.
BYTE	1 byte	Filled flag	TRUE if shape filled, FALSE otherwise.
BYTE	1 byte	Styled line flag	TRUE if styled line, FALSE otherwise.
DOUBLE	8 bytes	Stroke width	Stroke line width.
BYTE	1 byte	Start cap type	Start cap type (must be one of PSPStyleCapType).
BYTE	1 byte	Start cap multipliers flag	TRUE if use the following start cap multipliers, FALSE otherwise.
DOUBLE	8 bytes	Start cap width multiplier	Start cap width (x) multiplier.
DOUBLE	8 bytes	Start cap height multiplier	Start cap height (y) multiplier.
BYTE	1 byte	End cap type	End cap type (must be one of PSPStyleCapType).
BYTE	1 byte	End cap multipliers flag	TRUE if use the following end cap multipliers, FALSE otherwise.
DOUBLE	8 bytes	End cap width multiplier	End cap width (x) multiplier.
DOUBLE	8 bytes	End cap height multiplier	End cap height (y) multiplier.
BYTE	1 byte	Join type	Shape join type (must be one of PSPStyleJoinType).
DOUBLE	8 bytes	Miter limit	Shape miter limit.
	unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***
Paint Style Sub-Block	See Section 5.4.4	Line paint style	Paint style to use when outlining this shape. (See Section “5.4.4 : Paint Style Sub-Block.”)
Paint Style Sub-Block	See Section 5.4.4	Fill paint style	Paint style to use when filling this shape. (See Section “5.4.4 : Paint Style Sub-Block.”)
Line Style Sub-Block	See Section 5.4.5	Styled line	Styled line data to use when outlining this shape. (See Section “5.4.5: Line Style Sub-Block.”)

Polyline Shape Definition Chunk:

DWORD	4 bytes	Chunk size	Length of Polyline Shape Definition Chunk.
DWORD	4 bytes	Polyline node count	Number of nodes in the polyline shape (i.e., number of following Node Chunks).
	unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Polyline Node Chunk (first node):

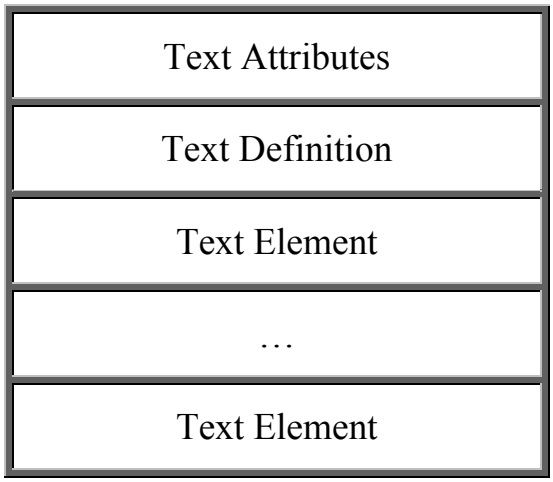
DWORD	4 bytes	Chunk size	Length of Polyline Node Chunk.
DOUBLE	8 bytes	Point X value.	X-value of node's main point.
DOUBLE	8 bytes	Point Y value	Y-value of node's main point.
DOUBLE	8 bytes	Handle 1 X value	X-value of node's first handle.
DOUBLE	8 bytes	Handle 1 Y value	Y-value of node's first handle.
DOUBLE	8 bytes	Handle 2 X value	X-value of node's second handle.
DOUBLE	8 bytes	Handle 2 Y value	Y-value of node's second handle.
BYTE	1 byte	Move to flag	TRUE if the node is a "move to" node, FALSE otherwise.
WORD	2 bytes	Node type flags	A series of flags (in PSPPolylineNodeTypes) that define the polyline node type.
	unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

• • •

Polyline Node Chunk (last node):

DWORD	4 bytes	Chunk size	Length of Polyline Node Chunk.
DOUBLE	8 bytes	Point X value.	X-value of node's main point.
DOUBLE	8 bytes	Point Y value	Y-value of node's main point.
DOUBLE	8 bytes	Handle 1 X value	X-value of node's first handle.
DOUBLE	8 bytes	Handle 1 Y value	Y-value of node's first handle.
DOUBLE	8 bytes	Handle 2 X value	X-value of node's second handle.
DOUBLE	8 bytes	Handle 2 Y value	Y-value of node's second handle.
BYTE	1 byte	Move to flag	TRUE if the node is a "move to" node, FALSE otherwise.
WORD	2 bytes	Node type flags	A series of flags (in PSPPolylineNodeTypes) that define the polyline node type.
	unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Text Vector Shape Definition:



The Text Vector Shape Definition consists of the chunks required to define one text shape. As illustrated below, the Text Vector Shape Definition consists of the Text Shape Attributes Chunk, the Text Shape Definition Chunk, and the Text Shape Element Chunk(s).

Text Shape Attributes Chunk:

DWORD	4 bytes	Chunk size	Length of Text Shape Attributes Chunk.
BYTE	1 byte	Text alignment	Text alignment (must be one of PSPTextAlignment).
LONG	4 bytes	X insert point	X value of the text insertion point.
LONG	4 bytes	Y insert point	Y value of the text insertion point.
DOUBLE	8 bytes	Deformation value #1	First element of the text deformation matrix. The text deformation matrix is a 3x3 matrix.
DOUBLE	8 bytes	Deformation value #2	Second element of the text deformation matrix.
DOUBLE	8 bytes	Deformation value #3	Third element of the text deformation matrix.
DOUBLE	8 bytes	Deformation value #4	Fourth element of the text deformation matrix.
DOUBLE	8 bytes	Deformation value #5	Fifth element of the text deformation matrix.
DOUBLE	8 bytes	Deformation value #6	Sixth element of the text deformation matrix.
DOUBLE	8 bytes	Deformation value #7	Seventh element of the text deformation matrix.
DOUBLE	8 bytes	Deformation value #8	Eight element of the text deformation matrix.
DOUBLE	8 bytes	Deformation value #9	Ninth element of the text deformation matrix.

BYTE	1 byte	Text Flow	Text flow, must be one of PSPTextFlow
DOUBLE	8 bytes	Path offset	For text on a path. This defines the offset from the text to the path.
	unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Text Shape Definition Chunk:

DWORD	4 bytes	Chunk size	Length of Text Shape Definition Chunk.
DWORD	4 bytes	Text element count	Number of text elements in the text shape (i.e., number of following Text Shape Element Chunks).
	Unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Text Shape Element (first text element):

The first text shape element (see "Text Shape Element Definition" below).

• • •

Text Shape Element (last text element):

The last text shape element (see "Text Shape Element Definition" below).

Text Shape Element Definition:

The Text Shape Element Definition consists of the chunks required to define one text shape element. As illustrated below, the Text Shape Element Definition consists of the Text Element Attributes Chunk and the Text Element Definition Chunk.

Text Element Attributes Chunk:

DWORD	4 bytes	Chunk size	Length of Text Element Attributes Chunk.
WORD	2 bytes	Text element type	Type of text element (must be one of PSPTextElementType).
	Unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Text Element Definition Chunk:

The contents of the Text Element Definition Chunk are based on the text element type (defined in the previous Text Element Attributes Chunk). If the text element type is keTextElemChar (a single character code), the contents are as described in the Text Character Definition Chunk (below). If the text element type is keTextElemCharStyle (a character style change), the contents are as described in the Text Character Style Definition Chunk (below). If the text element type is keTextElemLineStyle (a line style change), the contents are as described in the Text Line Style Definition Chunk (below).

Text Character Definition Chunk:

DWORD	4 bytes	Chunk size	Length of Text Character Definition Chunk.
DWORD	4 bytes	Character code	The character code (for a single character). This is the unicode value of the character.
	unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Text Character Style Definition Chunk:

DWORD	4 bytes	Chunk size	Length of Text Character Style Definition Chunk.
Variable length string chunk	See Section 5.4.2	Font name	Character font name. (Please See Section "5.4.2 – Variable Length String Chunk" for more information about variable string chunks.)
DWORD	4 bytes	Attribute flags	Character attribute flags (defined by PSPCharacterProperties).
DWORD	4 bytes	Weight	Character weight.
LONG	4 bytes	Set	Character set.
LONG	4 bytes	Font size	Size of font.
BYTE	1 byte	Anti alias mode	One of PSPAntialiasMode
BYTE	1 byte	Justify	Mode for text justify (left, center, right)
BYTE	1 byte	Auto kerning flag	TRUE if auto kern, FALSE otherwise.
DOUBLE	8 bytes	Kerning	Text kerning value.
DOUBLE	8 bytes	Tracking	Text tracking value.

DOUBLE	8 bytes	Leading	Text leading value
BYTE	1 byte	Stroked flag	TRUE if stroked, FALSE otherwise.
BYTE	1 byte	Filled flag	TRUE if shape filled, FALSE otherwise.
BYTE	1 byte	Styled line flag	TRUE if styled line, FALSE otherwise.
DOUBLE	8 bytes	Stroke width	Stroke line width.
BYTE	1 byte	Start cap type	Start cap type (must be one of PSPStyleCapType).
BYTE	1 byte	Start cap multipliers flag	TRUE if use the following start cap multipliers, FALSE otherwise.
DOUBLE	8 bytes	Start cap width multiplier	Start cap width (x) multiplier.
DOUBLE	8 bytes	Start cap height multiplier	Start cap height (y) multiplier.
BYTE	1 byte	End cap type	End cap type (must be one of PSPStyleCapType).
BYTE	1 byte	End cap multipliers flag	TRUE if use the following end cap multipliers, FALSE otherwise.
DOUBLE	8 bytes	End cap width multiplier	End cap width (x) multiplier.
DOUBLE	8 bytes	End cap height multiplier	End cap height (y) multiplier.
BYTE	1 byte	Join type	Shape join type (must be one of PSPStyleJoinType).
DOUBLE	8 bytes	Miter limit	Shape miter limit.
Paint Style Sub-Block	See Section 5.4.4	Line paint style	Paint style to use when outlining this shape. (See Section “5.4.4 : Paint Style Sub-Block.”)
Paint Style Sub-Block	See Section 5.4.4	Fill paint style	Paint style to use when filling this shape. (See Section “5.4.4 : Paint Style Sub-Block.”)
Line Style Sub-Block	See Section 5.4.5	Styled line	Styled line data to use when outlining this shape. (See Section “5.4.5: Line Style Sub-Block.”)
	unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

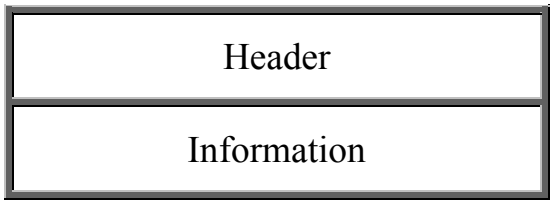
Group Vector Shape Definition:

The Group Vector Shape indicates the start of a group of shapes. The number of shapes in this group is defined in the Group Shape Definition Chunk. As illustrated below, the Group Vector Shape Definition consists only of the Group Shape Definition Chunk.

Group Shape Definition Chunk:

DWORD	4 bytes	Chunk size	Length of Group Shape Definition Chunk.
DWORD	4 bytes	Shape count	Number of shapes in this group (i.e., number of following Vector Shape Sub-Blocks belonging to this group).
	unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

6.11.3 Group Layer Sub-Block



The Group Layer Sub-Block contains layer extension data required to define a group layer.

As illustrated below, the Group Layer Sub-Block consists of the Group Layer Block Header and the Group Layer Information Chunk.

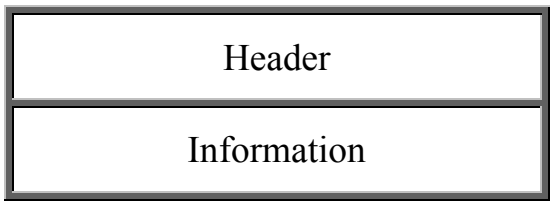
Group Layer Block Header:

Type	Length	Name	Description
B_ARRAY	4 bytes	Header identifier	Always "`7E 42 4B 00`" (i.e.,"~BK", followed by a zero byte).
WORD	2 bytes	Block identifier	Always PSP_GROUP_EXTENSION_BLOCK.
DWORD	4 bytes	Total block length	Length of complete Group Layer Sub-Block, including the Group Layer Information Chunk.

Group Layer Information Chunk:

DWORD	4 bytes	Chunk size	Length of Group Layer Information Chunk.
DWORD	4 bytes	Layer count	Number of layers in this group (does not include count of layers in sub-group layers).
BYTE	1 byte	Linked	TRUE if layer group is linked.
	Unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

6.11.4 Mask Layer Sub-Block



The Mask Layer Sub-Block contains layer extension data required to define a mask layer.

As illustrated below, the Mask Layer Sub-Block consists of the Mask Layer Block Header and the Mask Layer Information Chunk.

Mask Layer Block Header:

Type	Length	Name	Description
B_ARRAY	4 bytes	Header identifier	Always "`7E 42 4B 00`" (i.e.,"~BK", followed by a zero byte).
WORD	2 bytes	Block identifier	Always PSP_MASK_EXTENSION_BLOCK.
DWORD	4 bytes	Total block length	Length of complete Mask Layer Sub-Block, including the Mask Layer Information Chunk.

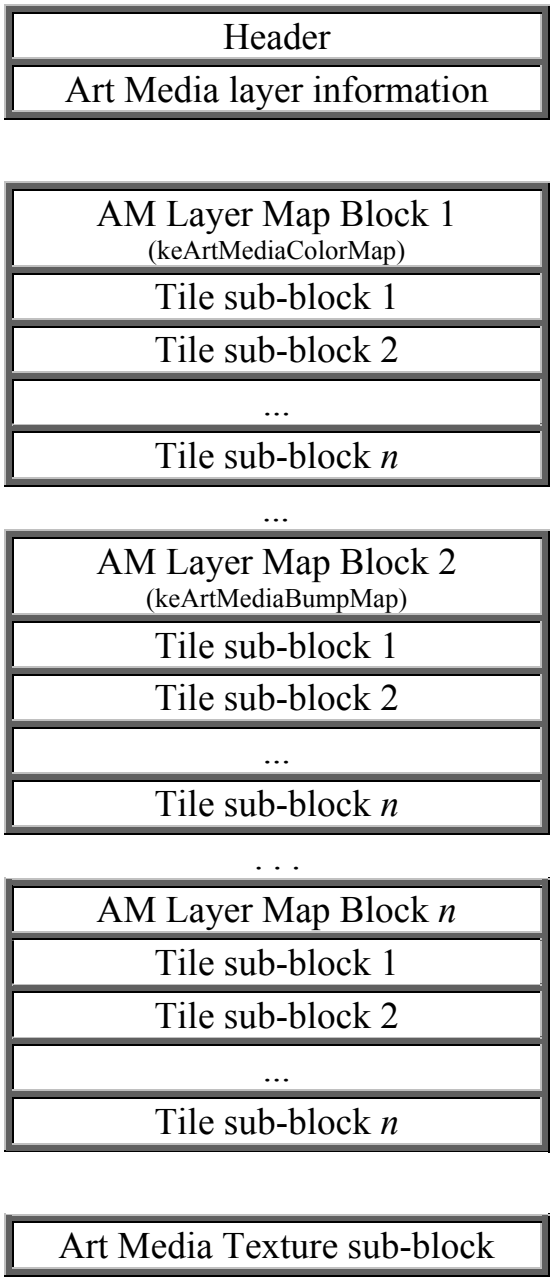
Mask Layer Information Chunk:

DWORD	4 bytes	Chunk size	Length of Mask Layer Information Chunk.
DWORD	4 bytes	Overlay color	Mask overlay color (RGB).
BYTE	1 byte	Opacity	Mask layer opacity (0 transparent - 100 opaque).
	Unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

6.11.5 Art Media Layer Sub-Block

The Art Media Layer Sub-Block contains layer extension data required to define an Art Media layer.

As illustrated below, the Art Media Layer Sub-Block consists of the Art Media Layer Block Header, Art Media Map sub-blocks and Art Media Tile sub-blocks. There are 5 Art Media Map sub-blocks defined by PSPArtMediaMapType, written in this order: keArtMediaColorMap, keArtMediaBumpMap, keArtMediaShininessMap, keArtMediaReflectivityMap, keArtMediaDrynessMap. Note that the data is written as sub-blocks and not chunks and that all 5 map blocks are written even if they contain no data. Within each map block are a series of tile blocks of size 128x128 that represent the the portion of the layer that contains art media data. Section 6.10.5.1 describes an Art Media Map sub-block. Section 6.10.5.2 describes an Art Media Tile sub-block.



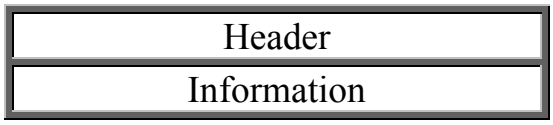
Art Media Layer Block Header:

Type	Length	Name	Description
B_ARRAY	4 bytes	Header identifier	Always "7E 42 4B 00" (i.e.,"~BK", followed by a zero byte).
WORD	2 bytes	Block identifier	Always PSP_ART_MEDIA_BLOCK.
DWORD	4 bytes	Total block length	Length of complete Art Media Layer Sub-Block, including the Art Media Layer Information Chunk.

Art Media Layer Information Chunk:

DWORD	4 bytes	Chunk size	Length of Art Media Layer Information Chunk.
WORD	16 bytes	Map count	Number of maps in layer
DWORD	4 bytes	Texture	Boolean value indicating if texture is present
BYTE	1 byte	Fill canvas	Boolean value indicating if canvas should be filled with a color
COLOR	3 bytes	Fill color	Specifies what color to fill with
DWORD	4 bytes	Sequence Number	Sequence number of stokes on this art media layer
	Unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

6.11.5.1 Art Media Layer Map sub-block



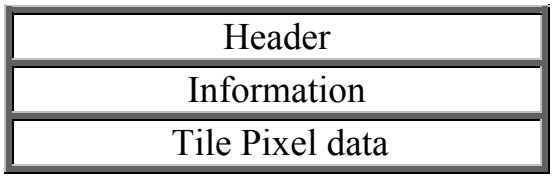
Art Media Layer Map sub-block Header:

Type	Length	Name	Description
B_ARRAY	4 bytes	Header identifier	Always "7E 42 4B 00" (i.e.,"~BK", followed by a zero byte).
WORD	2 bytes	Block identifier	Always PSP_ART_MEDIA_MAP_BLOCK.
DWORD	4 bytes	Total block length	Length of complete Art Media Layer Sub-Block, including the Art Media Layer Information Chunk.

Art Media Map Information Chunk:

DWORD	4 bytes	Chunk size	Length of Map Information Chunk.
RECT	16 bytes	Image rectangle	Rectangle containing selected data for this map
DWORD	4 bytes	Map type	one of PSPArtMediaMapType
DWORD	4 bytes	Memory method	allocation method, always 3
LONG x 2	8 bytes	Array Size	Size of the array of tiles (x,y)
LONG x 2	8 bytes	Tile Size	Size of each tile (x,y)
WORD	2 bytes	Compression	one of PSPCompression, either PSP_COMP_NONE or PSP_COMP_LZ77
	Unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

6.11.5.2 Art Media Tile sub-block



If the tile has no data, only the header will be written and not the Art Media Map Information chunk or pixel data. The pixel data is written after the information chunk, starting at row 0. Each row is then written in sequence using LZ77 compression. The data is written in different bit depths depending on the map type contained in the Art Media Map sub-block.

keArtMediaColorMap – 32 bits/pixel
keArtMediaBumpMap – 16 bits/pixel
keArtMediaShininessMap – 8 bits/pixel
keArtMediaReflectivityMap – 8 bits/pixel
keArtMediaDrynessMap – 16 bits/pixel

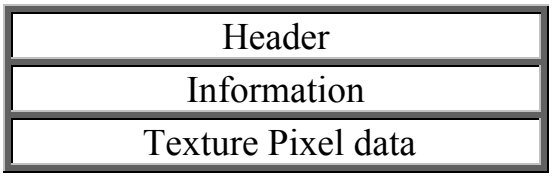
Art Media Tile Block Header:

Type	Length	Name	Description
B_ARRAY	4 bytes	Header identifier	Always "`7E 42 4B 00`" (i.e.,"~BK", followed by a zero byte).
WORD	2 bytes	Block identifier	Always PSP_ART_MEDIA_TILE_BLOCK.
DWORD	4 bytes	Total block length	Length of complete Art Media Tile Sub-Block, including the Art Media Tile Information Chunk.

Art Media Tile Information Chunk:

DWORD	4 bytes	Chunk size	Length of Tile Information Chunk.
RECT	16 bytes	Has tile	Does this tile have data?
DWORD	4 bytes	Bytes to copy	Bitmap width of current tile (pixel width * bpp)
	Unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

6.11.5.3 Art Media Texture sub-block



The texture pixel data is written after the texture information chunk encoding each row starting from row 0 using LZ77 compression method.

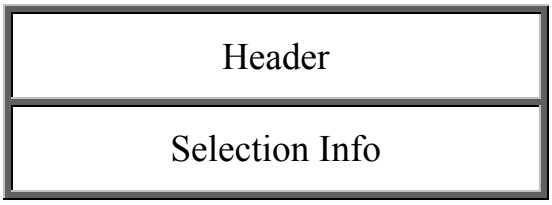
Art Media Texture sub-block Header:

Type	Length	Name	Description
B_ARRAY	4 bytes	Header identifier	Always "7E 42 4B 00" (i.e.,"~BK", followed by a zero byte).
WORD	2 bytes	Block identifier	Always PSP_ART_MEDIA_TEXTURE_BLOCK.
DWORD	4 bytes	Total block length	Length of complete Art Media Texture Sub-Block, not including the size of the pixel data

Art Media Texture Information Chunk:

DWORD	4 bytes	Chunk size	Length of Texture Information Chunk.
DWORD	4 bytes	Width	Width of texture bitmap in pixels
DWORD	4 bytes	Height	Height of texture bitmap in pixels
LONG x 2	8 bytes	Offset (x,y)	texture offset to preserve texture position when the layer is moved off canvas. If the layer is not moved off canvas this value will be (0,0). This value corresponds to a windows POINT structure
B_ARRAY	256 bytes	Name	Name of the texture, zero terminated string
	Unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

6.12 The Selection Block



Selection Channels

The Selection Block is an optional block that defines the area that was selected when the image document was saved.

As illustrated in the following table, the Selection Block consists of a Block Header, a Selection Information Chunk, and a Selection Channel Sub-Block.

Selection Header:

Type	Length	Name	Description
B_ARRAY	4 bytes	Header identifier	Always "`7E 42 4B 00`" (i.e.,"~BK", followed by a zero byte).
WORD	2 bytes	Block identifier	Always PSP_SELECTION_BLOCK.
DWORD	4 bytes	Total block length	Length of complete Selection Block, including Selection Information Chunk and Selection Channel Sub-Chunk.

Selection Information Chunk:

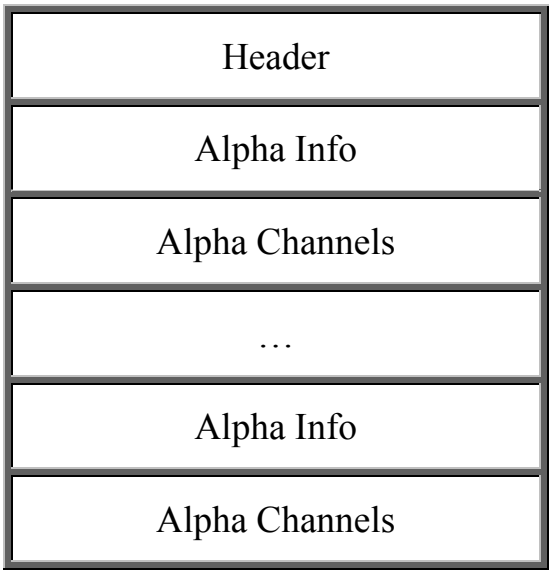
DWORD	4 bytes	Chunk size	Length of Selection Information Chunk.
RECT	16 bytes	Selection rectangle	Rectangle containing selected data.
RECT	16 bytes	Saved selection rectangle	Rectangle within selection rectangle that contains "significant" data. (Only the contents of this rectangle are saved to the file.)
	Unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Selection Channel Sub-Chunk:

DWORD	4 bytes	Chunk size	Length of Selection Channel Chunk.
WORD	2 bytes	Selection bitmap count	Count of following selection bitmaps. Currently always 1.
WORD	2 bytes	Channel count	Count of following channels. Currently always 1.
	Unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Selection Channel Sub-Block. See Section "5.4.8: Channel Sub-Block and Channel Compression." Note that selections are stored as 8-bit greyscale bitmaps.

6.13 The Alpha Bank Block



The Alpha Bank Block is an optional block that defines alpha channels associated with the image document.

As is illustrated in the following table, the Alpha Bank Block consists of a Block Header, an Alpha Channel Information Chunk, and an Alpha Channel Sub-Block for each alpha channel in the image document. (Each Alpha Channel Sub-Block consists of a Block Header and a Channel Sub-Block.)

Alpha Channel Bank Header:

Type	Length	Name	Description
B_ARRAY	4 bytes	Header identifier	Always "`7E 42 4B 00`" (i.e.,"~BK", followed by a zero byte).
WORD	2 bytes	Block identifier	Always PSP_ALPHA_BANK_BLOCK.
DWORD	4 bytes	Total block length	Length of complete Alpha Bank Block, including Alpha Bank Information Chunk, Alpha Channel Headers, Alpha Channel Information Chunks, and Alpha Channel Sub-Blocks for all alpha channels.

Alpha Bank Information Chunk:

DWORD	4 bytes	Chunk size	Length of Alpha Bank Information Chunk.
WORD	2 bytes	Alpha channel count	Number of alpha channels.
	Unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Alpha Channel Header (first alpha channel):

Type	Length	Name	Description
B_ARRAY	4 bytes	Header identifier	Always "`7E 42 4B 00`" (i.e.,"~BK", followed by a zero byte).
WORD	2 bytes	Block identifier	Always PSP_ALPHA_CHANNEL_BLOCK.
DWORD	4 bytes	Total block length	Length of complete Alpha Channel Block, including following Alpha Channel Information Chunk and Alpha Channel Sub-Block.

Alpha Channel Information Chunk (first alpha channel):

DWORD	4 bytes	Initial data chunk length	Length of Alpha Channel Information Chunk.
Variable length string chunk	See Section 5.4.2	Alpha channel name	Name of alpha channel. (Please See Section "5.4.2 – Variable Length String Chunk" for more information about variable string chunks.)
RECT	16 bytes	Alpha channel rectangle	Rectangle containing alpha channel.
RECT	16 bytes	Saved alpha channel rectangle	Rectangle within alpha channel rectangle that contains "significant" data. (Only the contents of this rectangle are saved to the file.)
	Unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Alpha Channel Chunk (first alpha channel):

DWORD	4 bytes	Chunk size	Length of Alpha Channel Chunk.
WORD	2 bytes	Alpha channel bitmap count	Number of following alpha channel bitmaps. (Currently always 1.)
WORD	2 bytes	Channel count	Number of following channels. (Currently always 1.)
	Unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Channel Sub-Block defining the alpha channel. See Section "5.4.8: Channel Sub-Block and Channel Compression." Note that alpha channels are stored as 8-bit greyscale bitmaps.

• • •

Alpha Channel Header (last alpha channel):

Type	Length	Name	Description
B_ARRAY	4 bytes	Header identifier	Always "7E 42 4B 00" (i.e.,"~BK", followed by a zero byte).
WORD	2 bytes	Block identifier	Always PSP_ALPHA_CHANNEL_BLOCK.
DWORD	4 bytes	Total block length	Length of complete Alpha Channel Block, including Alpha Channel Information Chunk and Alpha Channel Sub-Block.

Alpha Channel Information Chunk (last alpha channel):

DWORD	4 bytes	Initial data chunk length	Length of Alpha Channel Information Chunk.
Variable length string chunk	See Section 5.4.2	Alpha channel name	Name of alpha channel. (Please See Section "5.4.2 – Variable Length String Chunk" for more information about variable string chunks.)
RECT	16 bytes	Alpha channel rectangle	Rectangle containing alpha channel.
RECT	16 bytes	Saved alpha channel rectangle	Rectangle within alpha channel rectangle that contains "significant" data. (Only the contents of this rectangle are saved to the file.)
	Unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Alpha Channel Chunk (last alpha channel):

DWORD	4 bytes	Chunk size	Length of Alpha Channel Chunk.
WORD	2 bytes	Alpha channel bitmap count	Number of following alpha channel bitmaps. (Currently always 1.)
WORD	2 bytes	Channel count	Number of following channels. (Currently always 1.)
	Unknown	***Expansion field.***	***This field is currently non-existent, but robust readers should assume that data may be added here in the future, and silently ignore it.***

Channel Sub-Block defining the alpha channel. See Section "5.4.8: Channel Sub-Block and Channel Compression." Note that alpha channels are stored as 8-bit greyscale bitmaps.

Appendix A : PSP File Format Overview

PSP File Header	PSP File Signature
	PSP Major Version Number
	PSP Minor Version Number

General Image Attributes Block	General Image Attributes Block Header
	General Image Attributes

Extended Data Block	Extended Data Block Header
	• Keyword-Value
	• …
	• Keyword-Value

Color Profile Data Block	Color Profile Data Block Header
	Color Profile Data

Picture Tube Data Block	Picture Tube Data Block Header
	Picture Tube Data

Creator Data Block	Creator Data Block Header
	• Creator Key-Value
	• …
	• Creator Key-Value

Composite Image Bank Block	Composite Image Bank Block Header
	Composite Image Bank Info
	• Composite Image Attributes Block
	• …
	• Composite Image Attributes Block
	• Composite Image Block
	• …
	• Composite Image Block

Table Bank Block	Table Bank Block Header
	Table Bank Information
	• Table Block
	• …
	• Table Block

Color Palette Block	Color Palette Block Header
	Color Palette Entry Count
	• Color Palette Entry
	• …
	• Color Palette Entry

Layer Bank Block	Layer Bank Block Header
	• Layer Information • Layer Extension (if any) • Layer Channels
	• …
	• Layer Information • Layer Extension (if any) • Layer Channels

Selection Bank Block	Selection Bank Block Header
	Selection Info
	Selection Channels

Alpha Bank Block	Alpha Bank Block Header
	• Alpha Info • Alpha Channels
	• …
	• Alpha Info • Alpha Channels