Microsoft Windows

Maximum Performance from XPS Documents

October 2, 2007

Abstract

This paper describes how to create and process XPS documents efficiently. This paper will help application and printer-driver developers understand some of the more common issues and avoid the problems that can be encountered when creating XPS documents so that users of their applications can have the best customer experience.

This information applies for the following operating systems:

Windows Server® 2008
Windows Vista®
Microsoft® Windows Server 2003
Microsoft Windows® XP
Microsoft Windows 2000

What's new in this version: Minor changes to clarify optimization details presented in the original paper.

The current version of this paper is maintained on the Web at:

http://www.microsoft.com/whdc/xps/xpsdoc-perf.mspx

References and resources discussed here are listed at the end of this paper.

Contents

Disclaimer

This is a preliminary document and may be changed substantially prior to final commercial release of the software described herein.

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This White Paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Introduction

The XML Paper Specification describes a rich document-formatting language that makes it possible for you to save application content in a format that allows end users to see the document on a computer screen as they would see it when it is printed. The XML Paper Specification makes it possible to have this consistent document viewing experience on a wide range of operating systems, computer configurations, and printers and provides a range of storage and compression options that can be configured to provide the best user experience.

The richness of the feature set that the XML Paper Specification provides, however, can have some unintended consequences if applied incorrectly or inefficiently. The information in this paper will help you to provide end users with the best customer experience by showing you how to create the most efficient XPS documents from your application.

XPSDrv printer driver developers might also find the information about optimizing XPS document content to be useful. In some applications, printer driver performance might also be improved by optimizing the document content, markup, or packaging.

XPS Document Format

The XML Paper Specification describes the XPS document format and how it is organized internally and rendered externally. The Open Packaging Conventions describes how the parts of an XPS document are packaged for physical storage or transmission.

An XPS document is composed of modular parts and resources that are organized in a logical hierarchy. The format of the content and the organization of these parts and resources are specified in the XML Paper Specification. Figure 1 shows the parts of a sample XPS document in their logical hierarchy.

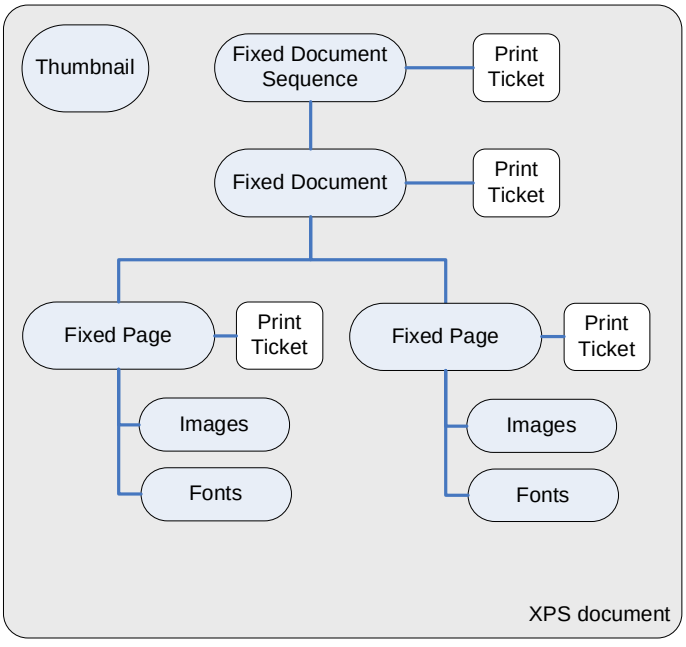

Figure 1. Logical Document-Part Hierarchy of an XPS Document

As an open specification, the XPS document format can be used to publish documents for end users who might be using computer systems and operating systems that are different from those that the author used to create the document. To be viewed, an XPS document does not require the original authoring application.

Although the XML Paper Specification describes the format for the content and the logical organization of the parts of an XPS document, the Open Packaging Conventions specifies the actual physical storage and organization of the document parts. It is important to understand this relationship because XPS documents can, and should, be optimized at both levels:

- The content and organization of the document parts that make up an XPS document can be optimized to represent the document content most efficiently.
- The physical storage can be managed and organized at the packaging layer so that the physical storage, transmission, and processing of the document can be made as efficient as possible.

This paper describes how you can create more efficient XPS documents from your application at both of these levels. In general, to provide the best document quality, the best end-user experience, and the most efficient document files at the same time, you should:

- Make the document markup as efficient as possible.
- Use the image format and compression level that provides the best image in the smallest space and still meets the requirements of the document. After the image has been compressed, do not compress it any further in the packaging layer. Compressing a compressed image reduces the overall file size but adds additional processing cost.
- Use font subsetting.
- Use the packaging layer compression options to compress the text and document markup in the packaging layer.

Performance and Optimization

In general, to optimize the performance of something usually means to optimize it for a specific scenario. With XPS documents, improving the performance of one parameter can sometimes come at the cost of another parameter. Therefore, it is important to consider the complete life cycle of the document before implementing any specific optimization. In some cases, the application might be required to present the document authors with a choice of optimization options so that they can provide the context that is necessary for the application to determine the best optimization scheme for a specific document or document scenario.

Some of the common performance trade-offs that you might encounter when deciding how to produce or process XPS documents include:

- **Processing speed versus storage size.** XPS documents can be compressed in several different ways. Compressing a document reduces the size of the overall document, which reduces the space required to store it and the communication bandwidth necessary to transfer it. Conversely, a highly compressed document requires more processing resources to manipulate, process, and display than a less compressed document. Which option is best depends on which factor is more critical.
- **Printing versus online display.** The parts of an XPS document can be organized at the packaging level so that they are optimized for an online viewer or print processing. An online viewing application, for example, might want to

display the text of a document as quickly as possible and render the images after the text. This would enable the end user to start reading the document text as quickly as possible and provides a better user experience. Conversely, a printer requires the text, the fonts, and the images of a page before it can render the page image on the paper. Because these two methods of document output process the document contents differently, the optimization for each situation is different.

- **Wait now or wait later.** The time at which a processing step occurs in the overall life cycle of a document may also be important. For example, when an author wants to view a draft version of a document, it might be better to render the document quickly instead of analyzing the content to make the document more efficient. However, when publishing to a large audience, it might be better for a document author to wait for the application to render the document most efficiently for the audience so that the end users have the best viewing or printing experience.

Maximizing XPS Document Performance

This section looks at some of the more common ways to improve XPS document efficiency when you create or process XPS documents. Some optimizations described in this section apply to every XPS document, whereas others might require user input to determine the best optimization technique. It might be better to perform some optimizations when spooling the document for printing to a printer, but others might be better if performed when saving the document as a file or for delivery over a network. Also, it might be worthwhile to apply some optimizations only to larger documents and not to smaller ones.

Creating Efficient Markup

Markup is the XML code that is used to describe the content and layout of an XPS document. How this markup is formatted can influence both the document size and the processing efficiency of the document. Markup that is used to describe objects that will not be rendered, such as when an object is clipped or covered by another opaque object, only adds to the processing overhead and overall size of the document without adding to the visible image. Eliminating this unused markup saves both space and processing. If a complex, layered image can be simplified or flattened, then the markup of that image can be simplified to save both space and processing time.

Flattening an image refers to merging layered image elements into a single, composite image. Because the XPS document format is used primarily as a *publishing* format, as opposed to a *working* format, the individual layers of an image or document that are present in the application are not necessary, so those layers can be merged into a single image layer to simplify the markup and reduce the size of the XPS document.

A *working* document format is used by the original, source document that the author created. In addition to the final text and images, it might also contain data that is useful to the author and the authoring process but that will not appear in the published document. Some examples of this type of information include the layers of image elements that were used to create the images and information such as change marks, revision histories, and editorial comments. This type of data would be used during the authoring process but would not be necessary in the published document.

A *publishing* document format contains only the information that is necessary to produce the finished, visible document content and layout. The content of a publishing document can often be greatly simplified by limiting the document content to only that which the end user sees.

Some of the document processing steps that can make a published XPS document more efficient are described in the following sections. When you consider how you will simplify the markup, be sure to consider the processing and the storage consequences of each step so that you get a net increase in document efficiency.

Reducing the Number of Element

Reducing the number of elements in the document part can make the document smaller and improve the subsequent processing of the document content, especially in resource-constrained environments. You can do this in your application if you:

- Flatten images that are made up of layers of elements into a single layer.
- Merge simple paths into a single path.
- Merge glyphs that have the same font, emSize, and OriginY attributes and that are close enough together horizontally into a single glyph. This is more efficient that creating one glyph per character.
- Use the simplest possible syntax in the element.
- Remove any hidden data that was used while authoring the document.
- Remove clipping elements that do not alter the image.
- Remove objects that will not be visible after clipping.
- Remove objects that have an opacity less than or equal to zero.
- Use a resource dictionary element to describe common components that can be reused.
- Use transformations to reuse path geometries that are stored in a resource dictionary by resizing and repositioning the stored objects as needed.
- Save complex brushes, such as those with a gradient or an image, in a resource dictionary if they are used more than once.
- Make complex brushes reusable so they can be stored in a resource dictionary and then use transforms as needed.
    - Change the StartPoint and EndPoint of a linear gradient brush to be within [(0,0)...(1,1)].
    - Change the Center, GradientOrigin, and Radius of a radial gradient brush to be within [(0,0)...(1,1)].
    - Change the Viewport of an image brush to be [0,0,1,1].
- Use a solid color brush instead of an image brush if the image brush is a uniform color.
- Use gradient brushes instead of a series of low-level calls that simulate a gradient.
- If the image is the same color vertically or horizontally, convert the image to a single column or row and repeat the row to fill the desired image size.

Simplifying Elements

Elements can be simplified by reducing the number of attributes and shortening the contents. This can help reduce the overall size and processing overhead as well. You can simplify the elements if you:

- Do not include default attributes. For example, if the default value for Opacity is 1.0 and the Opacity value for an element is 1.0, you can simplify that element by omitting the Opacity attribute.
- Group elements that have identical attributes into a single Canvas element. You can group elements that have identical Opacity, OpacityMask, Clipping, and Transformation attributes into a single Canvas element that has those attributes. You can then omit those attributes from each of the elements. Note: if the Opacity is not 1.0, you must consider how those elements overlap.
- Review section 5.1.10, "Optimizing Glyph Markup," of the XML Paper Specification to see if you can omit the glyph indices from your markup.
- Use shorter font uniform resource identifiers (URIs) for font resources that are not obfuscated.

Reducing and Simplifying Document Rendering Requirements

To further simplify the markup and reduce the processing that is required to render the document, you can:

- Use an integer coordinate space. If you don't need the 7-digit decimal precision of single-precision floating-point numbers, you can select a fixed integer resolution for your document, such as 600 dots per inch (DPI), and allow the document elements to be processed by using integer arithmetic instead of floating-point arithmetic.
- Use the simplest possible format to express geometry or a path. The more complex Path element is useful only when you have both Fill and Stroke attributes for the path and part of the geometry is not filled or stroked. Also, be sure that you don't include any unnecessary whitespace in the object.
- Remove any redundant points in the path unless they are the control points for Bezier curves.
- Use relative move commands.
- Consider using Bezier curves to describe small arcs. The Bezier curve may be more efficient to render. Also, use quadratic Bezier curves where possible. You could, for example, store a unit circle in a resource dictionary that was defined by four sections of Bezier curves. TrueType fonts are also stored internally as quadratic Bezier curves.
- Draw dashed lines by using the dash support for path elements instead of having a series of short paths.
- Consider using a VisualBrush to fill a large pattern instead of a series of simple geometry elements.
- Use the Opacity attribute instead of an OpacityMask where possible.
- Apply the Opacity attribute to the lowest element possible. For example, if a Canvas element has an Opacity attribute and has only one child element, remove the Opacity element from the Canvas element and apply it to the child element.
- Limit nested opacity values.

- Use image formats that support opacity rather than adding opacity to the image element for images with variable opacity. For example, a JPG image element with an opacity attribute could be converted to a PNG image that includes the specified opacity, reducing the resources required to render the image. For images with constant opacity, the Opacity element applied to an image without an alpha channel is easier to process than an image with an Alpha channel. Also consider image resolution and compression ratio to ensure file size is small and image quality is good. HD Photo has separate compression setting for color channels and alpha channel.
- Remove transformations that do not change the image such as <MatrixTransform Matrix="1,0,0,1,0,0" />.
- Reduce the clipping region to the page size unless a larger region is necessary such as when a larger area is necessary to support bleed areas.

Removing Redundant and Duplicate Resources

XPS documents support several types of resource parts. The most common resource parts are image resources and font resources. The default is to associate these resources with the FixedPage on which they are used; however, if an image or a font is used on several pages, this can lead to unnecessary duplication.

Image Resources

Normally the images that appear on a FixedPage are linked logically to that page. Sometimes the same image might be used on more than one page. Some common examples of images that are repeated in the document are a watermark image and a company logo. If a company logo appears on every page of a 100-page document, the document could include 100 identical image resources. The XML Paper Specification makes it possible to have the markup on each of the 100 pages with the image to reference the same image resource.

The watermark and logo examples are, perhaps, more extreme and obvious examples but, another, perhaps more subtle example might be when character elements are stored as image resources. This may be the result of a licensing or an application issue; however, referencing single copy of each image results in a greater space savings than storing a copy of the character image for every instance it is referenced in the document. Conversely, consolidating specific image or font references might slow processing through the XPSDrv printer driver filter pipeline.

XPS documents support HD Photo (also known as Microsoft® Windows® Media Photo) which uses an improved compression algorithm. Because HD Photo delivers compression quality comparable to JPEG-2000 and more than twice the quality of JPEG, consider using HD Photo for new images and applications. Storing image resources in this format can reduce document size while providing image quality that is comparable to other image formats. However, this savings is not usually worth the cost of converting existing image resources.

As mentioned earlier, large images tend to compress more efficiently when compressed as a single image than they do when divided into tiles. Consider keeping larger images together or merging images that are tiled by the application before compressing them for the published document.

Font Resources

Font subsetting can be used to reduce the size of font resources in an XPS document. Font subsetting is the XPS document feature that stores only the glyphs of a font that are used in the document instead of all the possible glyphs that the font supports. Some fonts, such as fonts that support more than one language or

character set, can contain a large number of glyphs. Without font subsetting, all of these glyphs are stored in the document even though the document may use only a small subset of them.

.NET Framework 3.0 makes it possible to adjust the scope of the subsetting from a single page to the entire fixed document sequence of an XPS document. At one extreme, adjusting the scope to include the entire fixed document sequence produces the smallest overall size of the font resource but can consume a large amount of system resources when preparing the document content to print. At the other extreme, setting the font subsetting scope to a single page increases the overall size of the font resources in the document, but might make it easier for the printer to process the document.

If an application can determine how the particular glyphs of each font are used in each page, it might be able to adjust this scope to achieve the best balance of size and processing performance. Glyph and font resources can be made more efficient if you:

- Replace fonts that use just a few glyphs with another format or convert those few glyphs to a vector or an image. This can reduce overall document size by replacing a font resource with a smaller image resource.
- Add individual glyph outlines or bitmaps to a resource dictionary and reuse the dictionary element if the font cannot be embedded.
- Merge glyphs that have the same font, emSize, and OriginY attributes and that are close enough together horizontally into a single glyph. This is more efficient that creating one glyph per character.

Optimizing File or Stream Size

At the most basic level, a larger document takes more time to move around than a smaller document. If all other factors are equal, just moving the bits of a larger document from one place to another takes longer than moving the bits of a smaller document. The perception of this difference depends on the throughput of the specific transport medium. Nevertheless, if your document can be made smaller, it can be moved faster.

When reducing the size, it is important to consider the processing effects of compression. Smaller documents might be faster to move than larger ones, but they might also take longer to process for printing or display. For example, if the file is compressed, it is smaller; however, if the file requires a lot of processing, the increase in the time required to uncompress, process, and recompress the document may negate any throughput gains. Conversely, a compressed document requires less storage space than an uncompressed document so, in an archive situation, the reduction in storage space might outweigh any increase in processing time.

How you weigh the different performance factors depends on the application, system configuration, and other situational factors, so this section cannot prescribe the single, best trade-off to make. Rather, it does present the different factors and provide you with the information that you can use to make your own determination.

Image Compression

In addition to removing redundant image resources, the type of compression that is used by the image resources in an XPS document affects the overall document size. For example, to maintain maximum image quality, tagged-image file format (TIFF) images are often not compressed when stored as an image resource. Using

a lossless compression algorithm for that image might reduce the size of the image resource while maintaining image quality. In other cases, a lossy compression algorithm could be used and could result in an even smaller image resource with only a minimal impact on the image fidelity.

Image compression algorithms tend to be more efficient, overall, when they compress a larger image as a single image than when the large image is subdivided into a series of smaller images or *tiles*. A single image that is compressed should be smaller than the sum of the compressed tiles that make up that image. If you must use a tiled image, use the tile brushes that the XML Paper Specification supports.

Text and Document Part Compression

The Open Packaging Conventions that is used by the XML Paper Specification employ ZIP compression on the document parts that contain text and markup. The degree to which the file size is reduced when the document parts are compressed depends on the content of the document parts and the type of compression used. Usually the processing resources to compress a document are not as much of a consideration as the processing resources required to send the document to end users and for those users to uncompress the document. In some cases, such as end users that have portable computers with limited processing capacity, it may be more effective to use documents that are not compressed. Conversely, when transmission bandwidth or storage capacity is limited, it might be more advantageous to use the maximum possible compression. Your application might provide the document author with the option to save the document as optimized for storage and transmission (smallest size) or for fastest processing (least compression).

Optimizing Document-Part Interleaving

Document interleaving refers to how the individual resource parts of an XPS document are streamed along with the fixed page document parts. .NET Framework 3.0 supports three types of interleaving orders:

- **ImagesLast.** The ImagesLast interleaving order streams text and font resources before the image resources. This interleaving order can help an online viewing application display the document text before rendering the images.
- **ResourceFirst.** The ResourceFirst interleaving order streams the resources before the document part in which they are used. This setting might be better for printers where the fonts and image resources are required to render the text.
- **ResourceLast.** The ResourceLast interleaving order streams the information about how a resource will be used before it streams the actual resource.

The ideal interleaving order to use when streaming an XPS document depends on how the document stream will be processed. For example, a document that will appear in a browser window might be better if it is streamed with the ImagesLast interleaving order. The ImagesLast interleaving order makes it possible for the browser display to render the document text before the images are processed so that end users can start to see document content more quickly.

Conversely, a document that is destined for a printer might print more quickly if it uses the ResourceFirst interleaving order. The ResourceFirst interleaving order would be good to use when spooling the document for printing because it allows the printer or the printer driver to cache the font resources when it starts processing the document so it can use them later when the document text is read. Putting the font resources at the end of the stream could possibly require the printer or the printer

driver to cache the entire document text before it could render the first page in the appropriate font. If an XPS-enabled printer has a limited memory, it might be worthwhile for the printer driver to ensure that the document stream to the printer has a ResourceFirst interleave order so that the printer can process the document parts most efficiently.

XPSDrv printer drivers have other streaming options that can be defined in the printer driver configuration file. For more information, see "XPSDrv Filter Pipeline: Implementation and Best Practices."

Chapter 10.1 of the XML Paper Specification discusses interleaving orders and how they can be optimized for different applications.

Using the DiscardControl Document Part

Chapter 10.1.3 of the XML Paper Specification describes how XPS document consumers can discard document parts to save resources in a resource-constrained environment. Chapter 10.1.4 of the XML Paper Specification describes the DiscardControl document part and how it can contain a list of the resources that the consumer can safely discard and when those document parts can be discarded. XPS document consumers can use the DiscardControl part to save additional resources while processing an XPS document. Each document package can contain only one DiscardControl part.

Using Banded Images

When processing images for printing, it is common for them to be *banded* by the printer driver before they are sent to the printer. Banding is the process of splitting an image or other graphical representation, such as a rasterized version of text, into horizontal bands. These bands are then sent to the printer device in sequence for printing. This process makes it possible to print large images on printers with limited memory or processing capacity.

Images should not be banded in XPS documents. The XML Paper Specification is not as strict as Graphics Device Interface (GDI) when joining the bands and so it is possible to have visible seams or breaks in a banded image when it is reconstructed from an XPS document. Also, banded images may not compress as efficiently as the complete image. It is still permissible for a printer driver to break an image into bands when it creates the page description language (PDL) or raster data to send to a printer that is not XPS enabled.

Rasterization versus Vector Graphics

Depending on how the document will be viewed or printed, it may be more efficient to render the document as rasterized content instead of vector-based graphics. The difference between the two versions depends on the complexity of the original vector graphics.

Call to Action

As an application or printer driver designer, you should provide the application users with the ability to use and configure the XPS document optimizations that will allow them to provide end users of the XPS documents with the best user experience.

Resources

XML Paper Specification
http://go.microsoft.com/fwlink/?LinkId=86085

October 2, 2007

Open Packaging Conventions (part of the Office Open XML document interchange specification)
http://go.microsoft.com/fwlink/?LinkId=86490

Print Schema Specification
http://go.microsoft.com/fwlink/?LinkId=86086

HD Photo Specification
http://go.microsoft.com/fwlink/?LinkId=86087

XPSDrv Filter Pipeline: Implementation and Best Practices
http://go.microsoft.com/fwlink/?LinkId=86489

Microsoft XML Paper Specification Essentials Pack
http://go.microsoft.com/fwlink/?LinkId=86088

Platform SDK
http://go.microsoft.com/fwlink/?LinkId=86089

Windows Driver Kit (WDK)
http://go.microsoft.com/fwlink/?LinkId=86090

XPS Home Page
http://go.microsoft.com/fwlink/?LinkId=86091

Windows Hardware and Device Central Home Page
http://go.microsoft.com/fwlink/?LinkId=86093