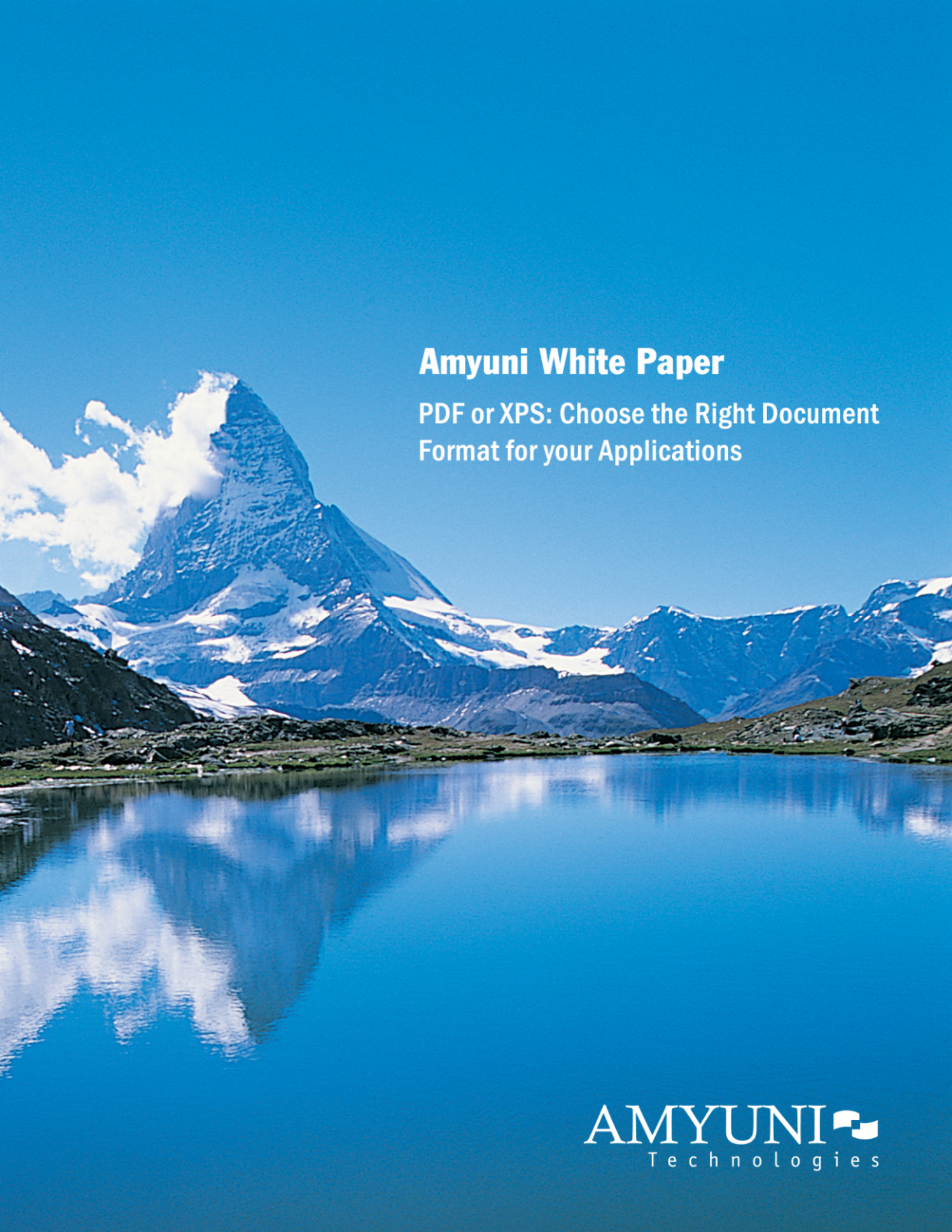


Amyuni White Paper

PDF or XPS: Choose the Right Document Format for your Applications

AMYUNI Technologies

PDF or XPS: Choose the Right Document Format for your Applications

By: Dany Amiouny
Amyuni Technologies

With the release of Windows Vista, Microsoft is introducing a new document format named "XML Paper Specification" or XPS. Microsoft describes XPS as follows:

"The XML Paper Specification (XPS) describes the format of a new general-purpose document made available by Microsoft to facilitate the easy exchange of documents between applications, platforms and hardware systems... XPS Documents offer a convenient alternative to paper documents for viewing, printing, transferring and archiving."[1]

The description looks a lot like what PDF does. The licensing of XPS is also very similar to the way Adobe licensed PDF in order to make it a widely used standard.

In this paper, we will look at the differences between PDF and XPS and advantages of each format. We will also look at ways to create either one of these formats and end up with some recommendations about which format to use in particular situations.

Originally, we had also looked at the Open Document Format supported by the OASIS group. But that format looked to us as being very archaic with a concept much different than PDF or XPS. This paper still includes a feature comparison table to compare the three formats but will mostly discuss PDF and XPS.

[1] http://www.microsoft.com/whdc/XPS/XpsLicense.mspx

Structure of a PDF File

A PDF file is made of a sequence of objects identified by numbers, followed by an index table containing the location of each object. Each object has a set of attributes associated with it. Some of the types of objects that can be found within a PDF file are: Document information, Page description, Font description, Colorspaces, Images, Form field objects, Annotations, …

Each object can have contents associated with it. Contents are usually compressed with flate compression (also known as ZIP), whereas object descriptions and attributes cannot be compressed.

The content of a page is a series of drawing instructions that define exactly how the page should view and print. Here's an example of a page description and its contents:

```
6 0 obj
<<
/Type /Page /Parent 3 0 R /MediaBox [0 0 612 792 ] /Contents [7 0 R ]
/Resources <</ProcSet [/PDF /Text]/Font <</F9 9 0 R >>>>
>>
endobj

7 0 obj
<< /Length 98 >>
stream
1 0 0 1 16.8 0 cm                          }
n                                          }
BT /F9 12  Tf 1 0 0 1 73.2 708.96 Tm       }  This part is usually compressed.
-0.075  Tc 0.435  Tw (Hello World) Tj      }
ET                                         }
endstream
endobj
```

In the example above, we're drawing text on a page using a specified font at a specific size and location. In PDF, a text fragment is not an object; it is a series of drawing instructions. The same thing applies to vector graphics and for some images.

This file format was designed for very fast viewing and printing. Its main inconveniences are:

- Extracting document contents in a meaningful way for analysis purposes can be very painful as the content is not structured.
- A larger file size because only fragments of the file are compressed.
- PDF files can be corrupted when sent by email or ftp because their content might be considered as text rather than binary.[2]

[2] http://www.amyuni.com/forum/viewtopic.php?p=1063

Note that Adobe made an attempt at compressing the whole file contents in its latest revisions of the format, but that resulted in backward incompatible files as we now have two different file formats for PDF.

Structure of an XPS File

An XPS file is made of a sequence of objects identified by URIs (Uniform Resource Identifiers.) The URI can be of the format /images/image1.jpg. The objects are stored in a regular ZIP file where all the object descriptions and object contents are compressed. The objects in a ZIP files are not indexed which makes access to an object much slower than in the case of PDF.

The content of a page is made of objects described in XML format rather than drawing instructions. Here's an example of a page description and contents in XPS:

```
<FixedPage Width="816" Height="1056"
xmlns="http://schemas.microsoft.com/xps/2005/06"
xml:lang="und">
<Glyphs Fill="#ff000000"
FontUri="/Documents/1/Resources/Fonts/71271D1B84C9.o
dttf" FontRenderingEmSize="15.9697"
StyleSimulations="None" OriginX="120"
OriginY="110.4"
Indices="43;72;79,27;79,29;82;3;58;82;85;79;71;3"
UnicodeString="Hello World " />
</FixedPage>
```

All this is compressed

Text here is an object that has various attributes and can be easily processed. The same thing applies to vector graphics and images.

This XML based format was designed to be easily extensible by simply adding attributes or objects to the schema. The main inconveniences of this format are:

- Once decompressed, the page content tends to be quite large.
- Processing page content can be much slower than PDF especially in printers that have more limited resources than PCs.
- Loading a complete document or searching for text within a document tends to be much slower than PDF.

Packaging XPS documents as a ZIP file makes the manipulation of XPS files much easier:

- Images and fonts are stored in their original format without the need for the extra processing that is required by PDF.
- The XML parts can be easily edited even with a simple text editor.

- One can easily add items to the ZIP package or replace some items. E.x. one can substitute an image with another by simply opening the ZIP package with standard tools. Note that this facility can be seen as a disadvantage or a security threat for some applications.

Benchmarks for Conversion Speed and File Size

In order to better evaluate the two file formats, we performed some benchmarks where we measured the conversion speed and the output file size in four different configurations:

1 – Using the XPS printer driver supplied by Microsoft.
2 – Using the Amyuni PDF Converter or printer driver product.[3]
3 – Using Adobe's Distiller product which is a PDF printer driver.
4 – Using a Postscript based PDF printer driver (there are hundreds of these on the market that are mostly based on the same technology and produce similar results.)

On the same PC with the same 156 page file:

Printer type	Conversion speed	File size
Microsoft XPS driver	6 seconds	1016 Kb
Amyuni PDF Converter	3 seconds	693 Kb
Adobe Distiller	9.6 seconds	846 Kb
Postscript based PDF driver	45 seconds	1413 Kb

More benchmarks from various applications and printers will be available on our web site.

It can be seen that using the Amyuni PDF printer, there would be a gain in performance and file size in using PDF versus XPS. Postscript based printer drivers are quite slow however due to the additional step of converting Postscript to PDF.

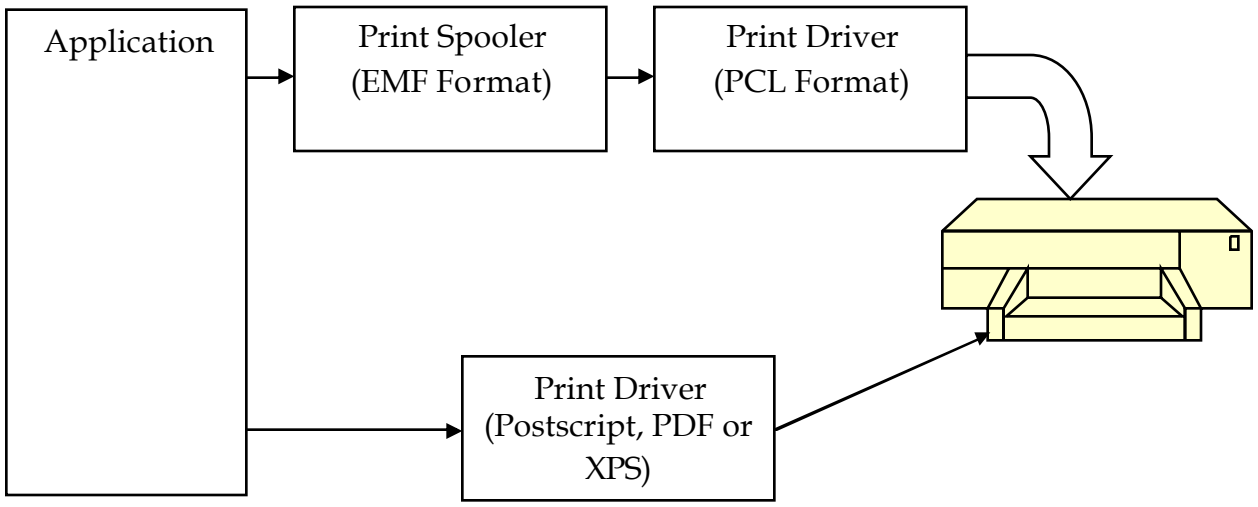
[3] Available from http://www.amyuni.com/en/developer/pdfconverter/features.html

XPS versus PDF as a Spool File Format

Starting with Windows Vista, XPS will become the format in which applications spool their data prior to sending it to a printer. This will replace EMF that had been in use since the early versions of Windows.

Some attempts have been made to spool PDF files directly to printers the same way Postscript files are sent to printers. PDF never gained much popularity in this area, mainly because the format was so complex and constantly changing that printer manufacturers preferred to stick to Postscript or PCL as their preferred printer language.

Microsoft has also redesigned the whole graphics subsystem in Vista to be based on XAML (the graphics format behind XPS.) Applications that are built using the WinFx framework can output XPS documents directly. Some printer manufacturers are also getting their printer engines ready to directly accept XPS files rather than having the additional step of converting XPS into their internal language such as PCL prior to printing.



XPS files contain an object named PrintTicket that contains all the information pertaining to the current print job. A sample print ticket would look like this:

```
<psf:ParameterInit name="psk:JobCopiesAllDocuments"><psf:Value
xsi:type="xsd:integer">1</psf:Value></psf:ParameterInit><psf:Feature
name="psk:PageMediaSize"><psf:Option
name="psk:NorthAmericaLetter"><psf:ScoredProperty
name="psk:MediaSizeWidth"><psf:Value
xsi:type="xsd:integer">215900</psf:Value></psf:ScoredProperty><psf:ScoredPro
perty name="psk:MediaSizeHeight"><psf:Value
xsi:type="xsd:integer">279400</psf:Value></psf:ScoredProperty></psf:Option><
```

```
/psf:Feature><psf:Feature name="psk:JobInputBin"><psf:Option
name="psk:AutoSelect"/></psf:Feature>
```

PrintTickets give XPS the ability to retain print job information. When the document is stored for printing at a later date, it can be guaranteed to print as it was originally intended. A common issue that users currently face with PDF is that they prepare documents with specific pages that should go to specific trays, duplex mode enabled and store the document in PDF format. When they print the document later on, they lose all information pertaining to tray selection and duplexing.

Creating Documents with Structured Content

PDF and XPS can be used to do more than simply view, print and archive documents. They can both be used as a general file format for storing all document types. This comes at a price however. Creating structured PDFs is not for the faint-hearted. Here's the concept of it.

Suppose you need to create a paragraph of text with a specific alignment and different text styles within the paragraph. A legitimate requirement would be to save the paragraph object in the PDF file and retrieve it as a paragraph when the file is opened. Because the contents of a page are only made of drawing instructions, you would usually retrieve a paragraph of text as a collection of unrelated text strings. The concept of tagged PDFs was devised by Adobe to help in that situation. It takes 80 pages of documentation to describe the concept, but here are the basic elements:

You start by surrounding the paragraph with the BDC/EMC tags such as:

```
/P << /MCID 0 >>
/BDC
BT /F9 12  Tf 1 0 0 1 73.2 708.96 Tm
-0.075  Tc 0.435  Tw (Hello World) Tj
ET
/EMC
```

Then you add an object in the PDF file outside of the page contents to set the attributes of the paragraph and on what page it is located:

```
4 0 obj
<< /Type /StructElem
/S /P /ID (Para1) /P 3 0 R /Pg 2 0 R /C /Normal
/A << /O /Layout /TextAlign /Justify >>
>>
endobj
```

Then you add to the page description a /StructParents that lists the paragraphs and other structured elements on the page. You will notice that as it happens very often in PDF, we obtain circular references where the page references the paragraph object and the paragraph references the page through the /Pg attribute.

Note that all the tags such as /P for paragraph are either predefined by Adobe or should be registered with Adobe to avoid conflicts.

In XPS, things are slightly simpler but much less documented. You start by creating Story elements in the document structure:

```
<DocumentStructure
xmlns="http://schemas.microsoft.com/xps/2005/06/documentstructure">
<Story Name="Story1">
<StoryFragmentReference FragmentName="FragmentA" Page="1"/>
</Story>
</DocumentStructure>
```

A Story contains StoryFragments which are a collection of named elements in the page contents:

```
<StoryFragments
xmlns="http://schemas.microsoft.com/xps/2005/06/documentstructure">
<StoryFragment StoryName="Story1" FragmentType="Paragraph">
<ParagraphStructure>
      <NamedElement NameReference="Para1" />
</ParagraphStructure>
</StoryFragment>
```

So the concepts are very similar, except that XPS does not have the issue with circular references that can make a developer's job a lot tougher.

Amyuni PDF Creator version 3.0[4] gives the developers the ability to create structured elements very simply by creating either a Table or a Group object. The table or group is given a name and other attribute values and will save and load in PDF or XPS as structured elements.

Amyuni PDF Creator will also convert non-structured PDF and XPS files into structured files. This is done by analyzing document content and applying various optimization steps to re-organize the page content into paragraphs and even tables.

[4] http://www.amyuni.com/en/developer/pdfcreator/features.html

Feature Comparison Table for PDF, XPS and ODF

This is a comparison table of document formats that are presented by their designers as being open document formats. The information below can serve as a basis for deciding which document format to use in your applications.

PDF : Portable Document Format by Adobe Corporation
XPS : XML Paper Specification by Microsoft Corporation
ODF : Open Document Format by the OASIS OpenDocument Committee

✓ Supported
☑ Support depends on version number
⊕ Poorly supported

General File Format	PDF	XPS	ODF
XML based extensible format		✓	✓
Portable / Multiplatform	✓	✓	✓
Full ZIP compression of file contents to reduce file size	☑ Compression at page level only	✓	✓
Fast page by page download from web servers	✓	✓	
Support for incremental file updates	✓		
Same file can contain multiple documents		✓	✓
Strong language enforcement (prohibit use of custom elements)	✓	✓	
Document Structure	**PDF**	**XPS**	**ODF**
Document bookmarks and outline	✓	✓	✓
Support for hyperlinks	✓	✓	✓
Support for page thumbnails	✓	✓	✓
Support for editable form fields	☑	⊕	✓
Support for annotations	✓	⊕	✓
Support for table objects	⊕	⊕	✓
Advanced Graphic Capabilities	**PDF**	**XPS**	**ODF**
Support for image transparencies	✓	✓	
Support for gradient fills	☑	✓	✓

Support for alpha channel in color definitions (opacity)		✓	✓
Editing Document Content	**PDF**	**XPS**	**ODF**
Ability to retrieve original document for editing in source application (e.g. Office)	⊕	⊕	✓
Ability to edit text in paragraphs with reflow	⊕	✓	✓
Support for text or paragraph styles			✓
Change tracking			✓
Security	**PDF**	**XPS**	**ODF**
Password protection against unauthorized access	✓	✓	✓
Password protection again document modification	⊕		
Digital signature of full document	✓	✓	
Digital signature of specific parts of a document	☑	✓	
Printing	**PDF**	**XPS**	**ODF**
Ensures document fidelity (document looks the same way it was intended)	✓	✓	
CMYK support for professional printing	✓	✓	
Retains print job information for deferred or remote printing		✓	
Support for crop boxes and bleed boxes needed by professional printers	✓	✓	
Image Support	**PDF**	**XPS**	**ODF**
JPEG (RGB and CMYK)	✓	✓	**Not enforced by format specifications**
JPEG 2000	☑		
JBIG2 for grayscale images	☑		
PNG images	☑	✓	
CCITT for black and white images	✓	⊕ Only within Tiff	
TIFF images (RGB and CMYK)		✓	
Windows Media Photo Images		✓	
Color Space Support	**PDF**	**XPS**	**ODF**
RGB	✓	✓	✓

CMYK	✓	✓	
Gray	✓	✓	
N-Channel	✓	✓	
ICC Profiles	✓	✓	
Named and Spot colors (e.g. Pantone)	✓	⊕	
Font Support	**PDF**	**XPS**	**ODF**
TrueType fonts	✓	✓	**Not enforced by format specifications**
Type1 (Postscript) fonts	✓		
CFF (Compact Font Format or Type2) fonts	✓	✓	
Vector fonts	✓		
Support for multi-national character sets	⊕ Unicode only	✓	⊕ Unicode only
Full font embedding	✓	✓	
Partial font embedding	✓	✓	
Protecting an embedded font from being extracted from the document	✓	⊕	
Recommended Applications	**PDF**	**XPS**	**ODF**
Remote document printing		✓	
Document archival	✓	✓	
Document storage on web servers for visitor downloads	✓		
Form filling / form based applications	✓		
Document exchange and collaboration			✓
General file format suitable for multiple application types		⊕	✓

Conclusion

XPS is bound to be a serious competitor to PDF. It has the advantage of having learned a lot from PDF without going through multiple iterations throughout the years. PDF still has first mover advantage. With the millions of PDF documents and hundreds of PDF tools out there, PDF is not going to disappear anytime soon.

XPS 1.0 has no support for form fields, scripting or any type of user interaction. These features will probably be included in future versions of XPS. Consequently, PDF will remain the most commonly used format for delivering fillable forms and form based applications.

In the meantime, developers will need to support two competing formats backed by two major players in the industry. Our aim at Amyuni Technologies is to provide a single source for creating, converting and processing both PDF and XPS documents using the same tools and the same interfaces.

As a recap, here's a list of our main tools and coming updates:

- Amyuni PDF Converter is a virtual printer driver that enables developers to create PDF documents with a few lines of code without doing any major changes to their existing applications. This is a high performance tool that is designed to do the job in a fraction of the time needed by other tools. Version 3.0 is under testing as of this writing and will support XPS the same way it supports PDF. For more details:
  http://www.amyuni.com/en/developer/pdfconverter/features.html

- Amyuni PDF Creator for .NET, ActiveX and Java is a set of libraries and controls that enable developers to process, create and print PDF documents. Version 3.0 is under testing as of this writing and will support XPS using the same interface without the need for the developers to rewrite their applications or integrate complex and expensive libraries. For more details:
  http://www.amyuni.com/en/developer/pdfcreator/features.html

About Amyuni Technologies

Amyuni Technologies is based in Montréal – Canada and specializes in document conversion, creation and processing. Amyuni Technologies took a leading position in PDF creation tools in 1999 and has since used its considerable experience in the PDF field to remain ahead of the competition.

Our comprehensive licensing scheme and product quality enabled developers to integrate our tools into hundreds of commercial applications installed on million of PCs worldwide.

For any comments, questions or queries, please contact management@amyuni.com