9. Parts and Relationships

The packaging conventions described in the OPC specification can be used to carry any payload. A *payload* is a complete collection of interdependent parts and relationships within a package. This specification defines a particular payload that contains a static or "fixed-layout" representation of paginated content: the fixed payload.

A package that holds at least one fixed payload and follows the rules described in this specification is referred to as an *XPS Document*. Producers and consumers of XPS Documents can implement their own parsers and rendering engines based on this specification.

XPS Documents address the requirements that information workers have for distributing, archiving, rendering, and processing documents. Using known rendering rules, XPS Documents can be unambiguously reproduced or printed without tying client devices or applications to specific operating systems or service libraries. Because the XPS Document is expressed in a neutral, application-independent way, the content can be viewed and printed without the application used to create the package.

9.1 Fixed Payload

A payload that has a FixedDocumentSequence root part is known as a *fixed payload*. A *fixed payload root* is a FixedDocumentSequence part that references FixedDocument parts that, in turn, reference FixedPage parts.

A specific relationship type is defined to identify the root of a fixed payload within an XPS Document: the *XPS Document StartPart relationship*. The *primary fixed payload root* is the FixedDocumentSequence part that is referenced by the XPS Document StartPart relationship. Consumers such as viewers or printers use the XPS Document StartPart relationship to find the primary fixed payload in a package. The XPS Document StartPart relationship MUST point to the FixedDocumentSequence part that identifies the root of the fixed payload [M2.14].

The payload includes the full set of parts required for processing the FixedDocumentSequence part. All content to be rendered MUST be contained in the XPS Document [M2.1]. The parts that can be found in an XPS Document are listed in Table 9–1. Relationships and content types for these parts are defined in §H. Each part MUST use *only* the appropriate content type specified in §H [M2.2].

Table 9–1. XPS Document parts

Name	Description	Required/Optional
FixedDocumentSequence (§9.1.2)	Specifies a sequence of fixed documents.	REQUIRED [M2.3]
FixedDocument (§9.1.3)	Specifies a sequence of fixed pages.	REQUIRED [M2.4]
FixedPage (§9.1.4)	Contains the description of the contents of a page.	REQUIRED [M2.5]
Font (§9.1.7)	Contains an OpenType or TrueType font.	REQUIRED if a <Glyphs> element is present [M2.6]

9.1.7 Font Parts

Fonts are stored in font parts. XPS Documents MUST support the OpenType font format, which includes TrueType and CFF fonts [M2.39]. To support portability, Unicode-encoded fonts SHOULD be used (see §9.1.7.5 for additional information) [S2.15].

Font parts are referenced using the FontUri attribute of the <Glyphs> element. A single font can be shared among multiple fixed pages in one or more fixed documents. Font references MUST be internal to the package; external references to fonts are invalid [M2.1].

If the referenced font part is a TrueType Collection, the fragment portion of the URI indicates the font face to be used. The use of URI fragments is specified in the BNF of Generic URI Syntax specification. The fragment contained in the FontURI attribute value MUST be an integer between 0 and n−1, inclusive, where n is the number of font faces contained in the TrueType Collection [M2.38]. The syntax for the integer value is expressed as:

```
fontface = *DIGIT
```

[*Example*: To reference the first font face in the font part "../Resources/Fonts/CJKSuper.ttc", the value of the FontUri attribute is "../Resources/Fonts/CJKSuper.ttc#0". *end example*] If no fragment is specified, the first font face is used in the same way as if the URI had specified "#0". If the fragment is not recognised as a valid integer, consumers SHOULD generate an error [S2.35].

Content types for fonts differ depending on whether the font is non-obfuscated or obfuscated (see §9.1.7.2). Content types are summarized in §H.

Fixed pages MUST use a Required Resource relationship to each Font parts referenced [M2.10]. For more information, see §H.

9.1.7.1 Subsetting Fonts

XPS Documents represent text using the <Glyphs> element. Since the format is fixed, it is possible to create a font subset that contains only the glyphs required by the package. Fonts MAY be subsetted based on glyph usage [O2.20]. Although a subsetted font does not contain all the glyphs in the original font, it MUST be a valid OpenType font file [M2.39]. Requirements for valid OpenType font files are described in the OpenType Font File specification.

9.1.7.2 OpenType Font Embedding

Protecting the intellectual property of font vendors is a goal of the XPS Document format. Therefore, producers MUST observe the guidelines and mechanisms described below in order to honor the licensing rights specified in OpenType fonts [M2.40]. It is not the responsibility of consumers to enforce font licensing intent, although consumers MUST be able to process XPS Documents using any combination of these embedding and obfuscation mechanisms, even if produced in violation of these guidelines [M2.41].

The licensing rights of an OpenType font are specified in the fsType field of the required OS/2 table in the font file. Table 9–7 lists the bit mask values that can appear in arbitrary combinations in the fsType field. Also listed are short descriptions of the licensing right intents and requirements or recommendations. These requirements represent the "rules" that producers and consumers must follow in order to respect licensing rights specified in the font.

For further details on licensing rights of OpenType fonts, see the description of the OS/2 table in "OS/2 and Windows Metrics."

Table 9–7. Guidelines for OpenType font embedding

Bit/mask	Licensing right intent	Producer rules	Consumer rules
– / 0x0000	Installable embedding.	SHOULD do embedded font obfuscation [S2.16] (see §9.1.7.3 for details).	SHOULD NOT extract or install permanently (see below) [S2.17].
0 / 0x0001	Reserved, must be 0.		
1 / 0x0002	Restricted license embedding. If *only* this bit is set, the font MUST NOT be modified, embedded or exchanged in any manner without obtaining permission from the legal owner.	MUST NOT embed [M2.42]. SHOULD generate a path filled with an image brush referencing an image of rendered characters [S2.18]. SHOULD include the text in the AutomationProperties.Name attribute of the <Path> element [S2.18].	Render embedded images.
2 / 0x0004	For preview and print embedding, font can be embedded and temporarily used on remote systems. However, documents containing *any* preview and print fonts MUST NOT be modified or edited [M2.43].	MUST do embedded font obfuscation [M2.44] (see §9.1.7.3). MUST add a Restricted Font relationship to the FixedDocument part of the document containing the font [M2.12]. See §12.1.7 and §H.3 for details.	MUST NOT extract or install permanently [M2.45]. MUST NOT modify or edit the XPS Document markup or hierarchical structure starting from the <FixedDocument> element [M2.43].
3 / 0x0008	Editable embedding.	MUST do embedded font obfuscation [M2.46] (see §9.1.7.3).	MUST NOT extract or install permanently [M2.47].
4–7	Reserved, must be 0.		
8 / 0x0100	No subsetting.	MUST do embedded font obfuscation (see §9.1.7.3) [M2.48]. MUST NOT subset font before embedding. [M2.49]	MUST NOT extract or install permanently [M2.50].
9 / 0x0200	Bitmap embedding only.	MUST do embedded font obfuscation [M2.51] (see §9.1.7.3). MUST embed *only* bitmap characters contained in the font [M2.51]. If no bitmap characters	MUST NOT extract or install permanently [M2.52].

	are present in the font, MUST NOT embed the font [M2.51].
10–15	Reserved, must be 0.

9.1.7.3 Embedded Font Obfuscation

Embedded font obfuscation is a means of preventing casual misappropriation of embedded fonts. Specifically, embedded font obfuscation prevents end-users from using standard ZIP utilities to extract fonts from XPS Document files and install them on their systems.

Embedded font obfuscation is *not* considered a strong encryption of the font data.

Embedded font obfuscation achieves the following goals:

1. Obfuscated font files are embedded within an XPS Document package in a form that cannot be directly installed on any client operating system.
2. Obfuscated font files are closely tied to the content referencing them. Therefore, it is non-trivial to misappropriate fonts by moving them from one package to another.
3. The manner in which obfuscated font files are tied to the content referencing them still allows for document merging.

For information on how to determine when fonts must be obfuscated prior to embedding, see Table 9–7. above.

Although the licensing intent allows embedding of non-obfuscated fonts and installation of the font on a remote client system under certain conditions, this is NOT RECOMMENDED in XPS Documents [S2.19]. However, there are vertical solutions in which implementations might benefit from un-obfuscated font embedding. In these cases, implementations could omit obfuscation or extract and install the embedded font.

If a producer is required to perform embedded font obfuscation, it MUST satisfy the following requirements [M2.53]:

1. Generate a 128-bit GUID (Globally Unique Identifier) for the font to be obfuscated. Instead of a true GUID, a 128-bit random number MAY be used [O2.21]. The 16 bytes of the 128-bit GUID are referred to in the following text by the placeholder names B_{00}, B_{01}, B_{02}, B_{03}; B_{10}, B_{11}; B_{20}, B_{21}; B_{30}, B_{31}, B_{32}, B_{33}, B_{34}, B_{35}, B_{36}, and B_{37}. The order in which bytes are assigned to these placeholders does not matter, as long as it is consistent for obfuscation and de-obfuscation.
2. Generate a part name for the obfuscated font using the GUID. The last segment of the part name MUST be of the form "$B_{03}B_{02}B_{01}B_{00}$-$B_{11}B_{10}$-$B_{21}B_{20}$-$B_{30}B_{31}$-$B_{32}B_{33}B_{34}B_{35}B_{36}B_{37}$" or "$B_{03}B_{02}B_{01}B_{00}$-$B_{11}B_{10}$-$B_{21}B_{20}$-$B_{30}B_{31}$-$B_{32}B_{33}B_{34}B_{35}B_{36}B_{37}$.ext" where each B_x represents a placeholder for one byte of the GUID, represented as two hex digits [M2.54]. The part name MAY have an arbitrary extension (identified by the placeholder ".ext") [O2.22]. It is RECOMMENDED that the extension for TrueType fonts be ".odttf" and for TrueType collections be ".odttc" [S2.20].
3. The content type for the part containing the obfuscated font MUST match the definition in §H [M2.2].
4. Perform an XOR operation on the first 32 bytes of the binary data of the font part with the array consisting of the bytes referred to by the placeholders B_{37}, B_{36}, B_{35}, B_{34}, B_{33},

B_{32}, B_{31}, B_{30}, B_{20}, B_{21}, B_{10}, B_{11}, B_{00}, B_{01}, B_{02}, and B_{03}, in that order and repeating the array once. The result is an obfuscated font.

5. Store the obfuscated font in a part with the generated name.

When processing fonts, consumers MUST follow these steps [M2.53]:

1. If the content type of the part containing the font is not the obfuscated font content type as specified in H, process the font without any de-obfuscation steps.
2. For font parts with the obfuscated font content type as specified in H, de-obfuscate the font by following these rules:~~0.~~2.
   a. Remove the extension from the last segment of the name of the part containing the font.
   b. Convert the remaining characters of the last segment to a GUID using the byte ordering described above.
   c. Perform an XOR operation on the first 32 bytes of the binary data of the obfuscated font part with the array consisting of the bytes referred to by the placeholders B_{37}, B_{36}, B_{35}, B_{34}, B_{33}, B_{32}, B_{31}, B_{30}, B_{20}, B_{21}, B_{10}, B_{11}, B_{00}, B_{01}, B_{02}, and B_{03}, in that order and repeating the array once. The result is a non-obfuscated font.
   d. Use the non-obfuscated font for the duration of the document processing, but do not leave any local or otherwise user-accessible copy of the non-obfuscated font.

9.1.7.4 Print and Preview Restricted Fonts

If a producer embeds a font with the print and preview restriction bit set, it MUST also add a Restricted Font relationship from the FixedDocument part that includes the FixedPage referencing the font to the restricted font [M2.12].

Consumers that are also producers MUST NOT edit a document where the FixedDocument part has a Restricted Font relationship [M2.43]. When invoking editing functionality, consumers that are also producers MUST treat as an error any font with the print and preview restriction bit set for which no Restricted Font relationship has been added to the FixedDocument part [M2.12].

Consumers that are not also producers MUST consider an XPS Document valid even if the producer failed to properly set the Restricted Font relationship [M2.12].

9.1.7.5 Non-Standard Font Compatibility Encoding

When processing <Glyphs> elements, the consumer MUST first select a cmap table from the OpenType font following the order of preference shown below (highest listed first) [M2.55]:

Table 9–8. Cmap table selection

Platform ID	Encoding ID	Description
3	10	Unicode with surrogates
3	1	Unicode without surrogates
3	5	Wansung
3	4	Big5
3	3	Prc
3	2	ShiftJis
3	0	Symbol

0	Any	Unicode (deprecated)
1	0	MacRoman

All further processing for that font MUST use the selected cmap table [M2.55].

If a Wansung, Big5, Prc, ShiftJis or MacRoman cmap has been selected, the consumer MUST correctly map from Unicode codepoints in the UnicodeString to the corresponding codepoints used by the cmap before looking up the glyphs [M2.56]. The Unicode standard provides details of the required mappings.

Producers SHOULD avoid using fonts lacking a Unicode-encoded cmap table [S2.15].

When processing <Glyphs> elements that reference a cmap (3,0) encoding font, consumers MUST be prepared for the case in which the UnicodeString attribute contains character codes instead of PUA codepoints [M2.57]. This condition is indicated by an unsuccessful Unicode lookup of the codepoint specified in the Unicode string in the cmap (3,0) table. In this case, the correct glyph index is computed by following the general recommendations of the OpenType specification.

When processing <Glyphs> elements that use this compatibility encoding, character codes in the range 0x20-0xff are mapped to PUA codepoints. Therefore, character codes in the range 0x80-0x9f are not considered non-printable Unicode control codes.

This non-standard encoding has been included to facilitate document production for certain producers. However, there are significant drawbacks resulting from this encoding:

- Search is unpredictable
- Copy and paste functionality is unpredictable

Producers SHOULD NOT use this non-standard encoding and they SHOULD write PUA codepoints to the UnicodeString attribute [S2.15].

9.1.8 Remote Resource Dictionary Parts

A *remote resource dictionary* allows producers to define resources that can be reused across many pages, such as a brush. This is stored in a Remote Resource Dictionary part. For more information, see §14.2.3.1.

9.1.9 PrintTicket Parts

PrintTicket parts provide user intent and device configuration information to printing consumers. PrintTicket parts MUST be processed when the XPS Document is printed [M2.58]. PrintTicket parts can be attached only to FixedDocumentSequence, FixedDocument and FixedPage parts and each of these parts MUST attach no more than one PrintTicket [M2.59]. PrintTickets can provide override settings to be used when printing the part to which they are attached.

9.1.9.1 PrintTicket Format

The PrintTicket is XML that provides print settings in a consistent, accessible, and extensible manner. Valid PrintTicket settings are specified in the Print Schema. Within the context of an XPS Document, the PrintTicket is generated by the producer. Producers should note that an XPS Document might be printed on various devices, and that the settings included in the PrintTicket SHOULD support portability [S2.21]. Producers and consumers should note that not all PrintTicket keywords defined in the Print Schema are applicable to XPS Documents.

9.2 Part Naming Recommendations

Producers and consumers of XPS Documents refer to parts by name and use relationship names to identify the purpose of related parts. The OPC specification describes the syntax for part names. However, following these rules alone can result in a package that is difficult for users to understand. [*Example*: A user would have to open every Relationship part to know which parts are necessary to accurately render an XPS Document. *end example*]

By choosing part names according to a well-defined, human-readable convention, the resulting package is easier to browse and specific parts are more easily located. Part names MUST still conform to the syntax specified in the OPC specification [M1.1].

It is RECOMMENDED that producers of XPS Documents use the following part naming convention:

- The FixedDocumentSequence part name SHOULD contain only one segment, and that segment SHOULD have the extension ".fdseq". [*Example*: "/FixedDocSeq.fdseq" *end example*] [S2.24].
- A FixedDocument part name SHOULD contain three segments, using "/Documents/*n*/" in the first two segments and the extension ".fdoc" [S2.25]. Here, *n* SHOULD be a numeral that represents the ordinal position of the fixed document in the fixed document sequence [S2.25]. [*Example*: The fixed document referenced by the Source attribute of the third <DocumentReference> child of the <FixedDocumentSequence> element could be "/Documents/3/FixedDocument.fdoc". *end example*]
- A FixedPage part name SHOULD contain four segments, using "/Documents/*n*/Pages/" as the first three segments and the extension ".fpage" on the last segment [S2.26]. Here, *n* represents the fixed document that includes this page. [*Example*: The third page of the second document might be "/Documents/2/Pages/3.fpage". *end example*]
- Resource parts MAY be named to indicate whether their intended use is at the document level or as a shared resource for all documents [O2.28]. A resource that is specific to a particular document SHOULD have a part name that begins with the three segments "/Documents/*n*/Resources/" where *n* is the particular fixed document [S2.27]. A resource intended to be shared across documents SHOULD begin with the segment "/Resources/" and SHOULD have a final segment that is a globally unique identifier followed by the appropriate extension for that resource [S2.27]. [*Example*: "/Resources/Fonts/63B51F81-C868-11D0-999C-00C04FD655E1.odttf" *end example*]

  A Font part name SHOULD append the segment "Fonts/" to the resource part name prefix specified above [S2.27]. [*Example*: A font might be named "/Documents/1/Resources/Fonts/Arial.ttf" or "/Resources/Fonts/F2ABC7B7-C60D-4FB9-AAE4-3CA0F6C7038A.odttf". *end example*]

  An Image part name SHOULD append the segment "Images/" to the resource part name specified above [S2.27]. [*Example*: An image might be named "/Documents/3/Resources/Images/dog.jpg" or "/Resources/Images/E0D79307-846E-11CE-9641-444553540000.jpg". *end example*]

  A Remote Resource Dictionary part name SHOULD append the segment "Dictionaries/" to the resource part name specified above [S2.27]. Remote resource dictionaries SHOULD also use the ".dict" extension [S2.27]. [*Example*: A resource dictionary might be named "/Documents/2/Resources/Dictionaries/Shapes.dict" or

"/Resources/Dictionaries/0DDF3BE2-E692-15D1-AB06-B0AA00BDD685.dict". *end example*]

- Any DocumentStructure part name SHOULD contain four segments using "/Documents/*n*/Structure/" as the first three segments and the extension ".struct" [S2.28]. Here *n* represents the fixed document that this structure is associated with. [*Example*: The DocumentStructure part for the first document in a fixed document sequence could be "/Documents/1/Structure/DocStructure.struct". *end example*]
- Any StoryFragments part name SHOULD contain five segments using "/Documents/*n*/Structure/Fragments" as the first four segments and the extension ".frag" [S2.29]. Here *n* represents the fixed document that these parts are associated with. [*Example*: A StoryFragment part associated with the third page of the second document in a fixed document sequence could be "/Documents/2/Structure/Fragments/3.frag". *end example*]
- ICC profile part names SHOULD contain four segments, using "/Documents/*n*/Metadata/" as the first three segments, where *n* is the fixed document that uses these parts [S2.30]. If an ICC profile part is shared across documents, the part name SHOULD contain two segments, using "/Metadata/" as the first segment and a second segment that is a string representation of a globally unique identifier, followed by an extension [S2.30]. ICC profiles SHOULD use an appropriate extension for the color profile type. [S2.30] [*Example*: ".icm" *end example*]
- Thumbnail part names SHOULD contain four segments, using "/Documents/*n*/Metadata/" as the first three segments, where *n* is the fixed document that uses the thumbnail [S2.31]. If the Thumbnail part relates to the package as a whole, the part name SHOULD contain two segments, using "/Metadata/" as the first segment and a second segment that is a string representation of a globally unique identifier, followed by an extension [S2.31]. Thumbnails SHOULD use an extension appropriate to the image type, either ".png" or ".jpg" [S2.31]. [*Example*: A Thumbnail part for a particular fixed page might be "/Documents/1/Metadata/5.png". *end example*]
- PrintTicket part names associated with the entire job SHOULD be associated via relationship with the FixedDocumentSequence part and contain two segments, using "/Metadata/" as the first segment [S2.32]. PrintTicket parts associated with a particular fixed document or fixed page SHOULD contain four segments, using "/Documents/*n*/Metadata/" as the first three segments, where *n* is the fixed document that uses these parts [S2.32]. PrintTicket parts SHOULD use the extension ".xml" [S2.32]. [*Example*: A PrintTicket associated with the entire job could be "/Metadata/Job_PT.xml" and a PrintTicket associated with a single page might be "/Documents/1/Metadata/Page2_PT.xml". *end example*]
- The names of any non-standard parts that are associated with a particular fixed document SHOULD contain four segments, using "/Documents/*n*/Other/" as the first three segments. Here, *n* is the fixed document to which the part belongs [S2.33].

Example 9–2. XPS Document part naming

An XPS Document that contains two FixedDocument parts is represented as follows:

```
/FixedDocSeq.fdseq
/Documents/1/FixedDocument.fdoc
/Documents/1/Pages/1.fpage
/Documents/1/Pages/2.fpage
/Documents/1/Resources/Fonts/FontA.ttf
/Documents/1/Resources/Images/ImageB.jpg
/Documents/1/Metadata/Document_PT.xml
```

```
/Documents/1/Metadata/Page5_PT.xml
/Documents/1/Structure/DocStructure.struct
/Documents/1/Structure/Fragments/1.frag
/Documents/1/Structure/Fragments/2.frag
/Documents/1/Other/FabrikamIncBussinessAccount.xml
/Documents/2/FixedDocument.fdoc
/Documents/2/Pages/1.fpage
/Documents/2/Resources/Fonts/FontB.ttf
/Documents/2/Resources/Images/ImageA.png
/Documents/2/Metadata/ColorProfile.icm
/Documents/2/Metadata/Document_PT.xml
/Documents/2/Other/FabrikamIncInsuranceInfo.xml
/Metadata/Job_PT.xml
/Resources/Fonts/63B51F81-C868-11D0-999C-00C04FD655E1.ttf
```

end example]

9.3 XPS Document Markup

XPS Document markup has been designed to facilitate the independent development of compatible systems that produce or consume XPS Documents. It also shares concepts with portions of the Microsoft .NET Framework 3.0 programming platform.

The graphics rendering model is shared with that of the Windows Presentation Foundation, assuring fidelity between on-screen display and printed output. The syntax of fixed page, fixed document, and fixed document sequence markup is compatible with that of Windows Presentation Foundation XAML. The elements, attributes, and attribute values are a subset of those defined by the Windows Presentation Foundation.

The relationship between XPS Document markup and .NET 3.0 technologies does not impose any requirement on system implementations. Support for XPS Document markup does not require incorporation of .NET 3.0, the Windows Presentation Foundation, or managed code. However, the relationship with .NET 3.0 technologies allows producers to extend XPS Document markup for further use in the Windows Presentation Foundation framework, such as by including additional presentation features.

The design of XPS Document markup reflects the tradeoffs between two, sometimes competing, goals:

1. XPS Document markup should be parsimonious; that is, it should include only the minimum set of primitive operations and markup constructs necessary to render text and graphics with full fidelity. Redundancy in the specification increases the opportunity for independent implementations, such as printer-resident raster image processors (RIPs), viewers, and interactive applications, to introduce accidental incompatibilities. Redundancy also increases the cost of implementation and testing, and, typically, the required memory footprint.
2. XPS Document markup should be compact; that is, the most common graphical primitives for vector graphics and text-rendering should have compact representations. Bloated representations compromise the performance of systems handling XPS Documents. As byte-count increases, so does communication time. Although compression can be used to improve communication time, it cannot eliminate the performance loss caused by bloated representations.

Example 9–4. Property element syntax

When specifying Clip and RenderTransform properties of the canvas, both must appear before any path and glyphs contents of the canvas.

```
<Canvas>
   <!-- First, the property-related child elements -->
   <Canvas.RenderTransform>
      <MatrixTransform Matrix="1,0,0,1,0,0" />
   </Canvas.RenderTransform>
   <Canvas.Clip>
      <PathGeometry>
         ...
      </PathGeometry>
   </Canvas.Clip>
   <!-- Then, the "contents" -->
   <Path ...>
      ...
   </Path>
   <Glyphs ... />
</Canvas>
```

end example]

9.3.4 Whitespace

XPS Documents allow flexible whitespace usage in markup. Wherever a single whitespace character is allowed, multiple whitespace characters MAY be used [O2.30]. Attributes that specify comma-delimited attribute values MAY, unless specified otherwise, OPTIONALLY include whitespace characters preceding or following the comma [O2.31]. XPS Document markup MUST NOT use the xml:space attribute [M2.75]. Additionally, where the XPS Document schema specifies attributes of types that allow whitespace collapsing, leading and trailing whitespace in the attribute value MAY be used along with other whitespace that relies on the whitespace collapsing behavior specified in the XML Schema Specification [O2.32].

[*Note*: Consult the XPS Document Schema for exact whitespace allowed. *end note*]

9.3.5 Language

Language information supports the following features:

- Language-dependent find features
- Selection of a text-to-speech dictionary by a screen-reading program (to provide accessibility to persons with disabilities)
- Selection of a spelling checker for text copied to another document
- Selection of a grammar checker for text copied to another document
- Correct font rendering when copying the text to another document

The last point refers to instances in which multiple languages share the same script. [*Example*: The Devanagari script is shared by the Indic languages Bhojpuri, Bihari, Hindi, Kashmiri, Konkani, Marathi, Nepali, and Sanskrit. However, these languages render certain glyph sequences differently. When text is copied from an XPS Document, the language of the copied characters is needed to ensure proper rendering of the glyphs when they are pasted into

another application. This scenario applies to most Indic-language fonts, some East Asian–language fonts, and others. *end example*]

9.3.5.1 xml:lang Attribute

The language of the contents of an XPS Document MUST be identified using the xml:lang attribute, the value of which is inherited by child and descendant elements [M2.76]. This attribute is defined in the W3C XML specification.

xml:lang is REQUIRED for <FixedPage> elements and MAY be used with <Canvas>, <Path>, and <Glyphs> elements; it is not valid on any other fixed page markup element [M2.72]. xml:lang is also REQUIRED for the <DocumentOutline> element for document structure and OPTIONAL for the <OutlineEntry> element [M2.72]. When the language of the contents is unknown and is required, the value "und" (undetermined) MUST be used [M2.76].

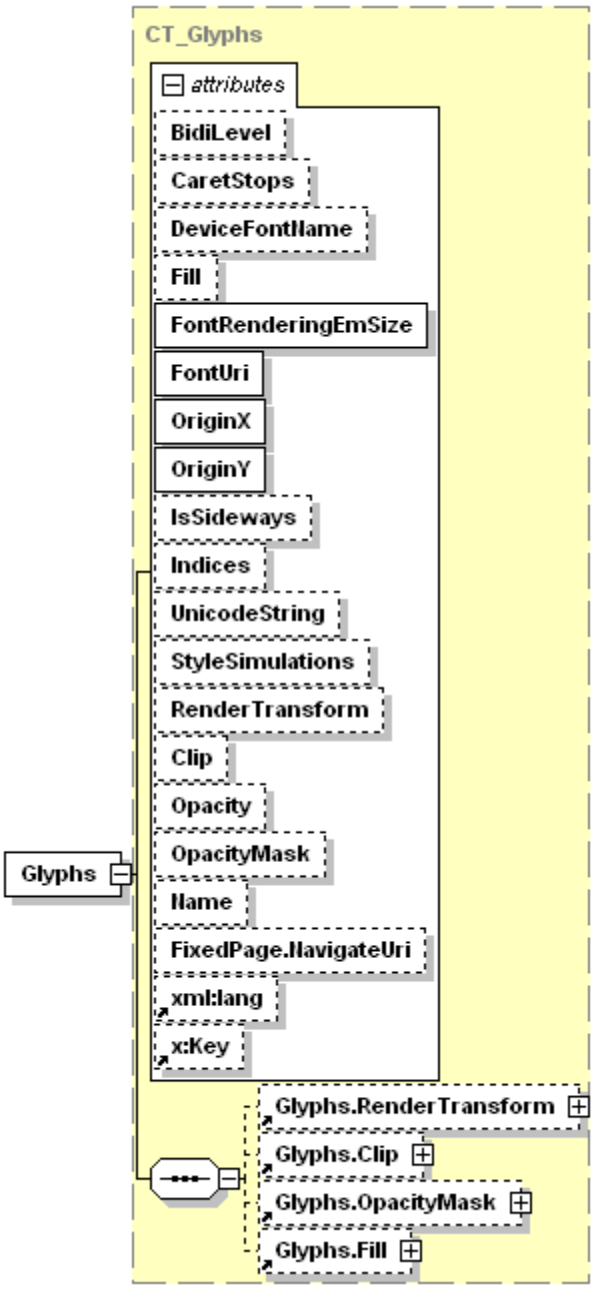
10.6 <Glyphs> Element

The <Glyphs> element is used to represent a run of uniformly-formatted text from a single font. The <Glyphs> element provides information for accurate rendering and supports search and selection features in XPS Document consumers. For more information, see §12.1.

12.1 <Glyphs> Element

element **Glyphs**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
BidiLevel			0		Specifies the Unicode algorithm bidirectional nesting level. Even values imply left-to-right layout,

						odd values imply right-to-left layout. Right-to-left layout places the run origin at the right side of the first glyph, with positive advance widths (representing advances to the left) placing subsequent glyphs to the left of the previous glyph. Valid values range from 0 to 61, inclusive.
	CaretStops	ST_CaretStops				Identifies the positions within the sequence of Unicode characters at which a text-selection tool can place a text-editing caret. Potential caret-stop positions are identified by their indices into the UTF-16 code units represented by the UnicodeString attribute value. When this attribute is missing, the text in the UnicodeString attribute value MUST be interpreted as having a caret stop between every Unicode UTF-16 code unit and at the beginning and end of the text [M5.1]. The value SHOULD indicate that the caret cannot stop in front of most combining marks or in front of the second UTF-16 code unit of UTF-16 surrogate pairs [S5.1].
	DeviceFontName	ST_UnicodeString				Uniquely identifies a specific device font. The identifier is typically defined by a hardware vendor or font vendor.
	Fill	ST_RscRefColor				Describes the brush used to fill the shape of the rendered glyphs.
	FontRenderingEmSize	ST_GEZero	required			Specifies the font size in drawing surface units, expressed as a float in units of the effective coordinate space. A value of 0 results in no visible text.
	FontUri	xs:anyURI	required			The URI of the physical font from which all glyphs in the run are drawn. The URI MUST reference a font contained in the package [M2.1]. If the physical font referenced is a TrueType Collection (containing multiple font faces),

					the fragment portion of the URI is a 0-based index indicating which font face of the TrueType Collection should be used.
	OriginX	ST_Double	required		Specifies the x coordinate of the first glyph in the run, in units of the effective coordinate space. The glyph is placed so that the leading edge of its advance vector and its baseline intersect with the point defined by the OriginX and OriginY attributes.
	OriginY	ST_Double	required		Specifies the y coordinate of the first glyph in the run, in units of the effective coordinate space. The glyph is placed so that the leading edge of its advance vector and its baseline intersect with the point defined by the OriginX and OriginY attributes.
	IsSideways	ST_Boolean		false	Indicates that a glyph is turned on its side, with the origin being defined as the top center of the unturned glyph.
	Indices	ST_Indices			Specifies a series of glyph indices and their attributes used for rendering the glyph run. If the UnicodeString attribute specifies an empty string ("" or "{}") and the Indices attribute is not specified or is also empty, a consumer MUST generate an error [M5.2].
	UnicodeString	ST_UnicodeString			Contains the string of text rendered by the <Glyphs> element. The text is specified as Unicode code points.
	StyleSimulations	ST_StyleSimulations		None	Specifies a style simulation. Valid values are None, ItalicSimulation, BoldSimulation, and BoldItalicSimulation.
	RenderTransform	ST_RscRefMatrix			Establishes a new coordinate frame for the glyph run specified by the <Glyphs> element. The render transform affects clip, opacity

					mask, fill, x origin, y origin, the actual shape of individual glyphs, and the advance widths. The render transform also affects the font size and values specified in the Indices attribute.
	Clip	ST_RscRefAbbrGeomF			Limits the rendered region of the element. Only portions of the <Glyphs> element that fall within the clip region (even partially clipped characters) produce marks on the page.
	Opacity	ST_ZeroOne		1.0	Defines the uniform transparency of the glyph element. Values range from 0 (fully transparent) to 1 (fully opaque), inclusive. Values outside of this range are invalid.
	OpacityMask	ST_RscRef			Specifies a mask of alpha values that is applied to the glyphs in the same fashion as the Opacity attribute, but allowing different alpha values for different areas of the element.
	Name	ST_Name			Contains a string value that identifies the current element as a named, addressable point in the document for the purpose of hyperlinking.
	FixedPage.NavigateUri	xs:anyURI			Associates a hyperlink URI with the element. May be a relative reference or a URI that addresses a resource that is internal to or external to the package.
	xml:lang				Specifies the default language used for the current element. The language is specified according to RFC 3066.
	x:Key				Specifies a name for a resource in a resource dictionary. x:Key MUST be present when the current element is defined in a resource dictionary. x:Key MUST NOT be specified outside of a resource dictionary [M5.3].

annotation	Represents a run of text from a single font.

The <Glyphs> element represents a run of uniformly-formatted text from a single font. It provides information necessary for accurate rendering and supports search and selection features in viewing consumers.

If the Fill property is not specified, the <Glyphs> element has no visible effect.

Some properties of the <Glyphs> element are composable, meaning that the markings rendered to the page are determined by a combination of the property and all the like-named properties of the <Glyphs> element's parent and ancestor elements. For details, see §14.

12.1.1 Glyph Metrics

Each glyph defines metrics that specify how it aligns with other glyphs. The metrics are illustrated below.

Figure 12–1. Glyph metrics

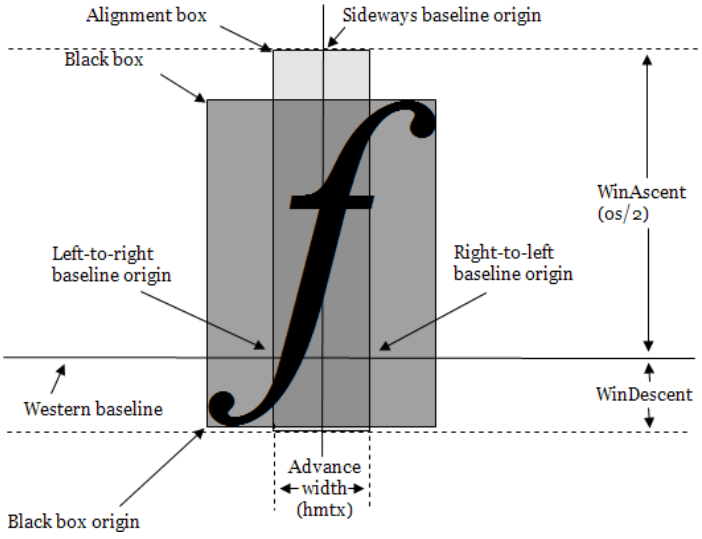


Figure 12–2. Upright (usually horizontal) glyph metrics

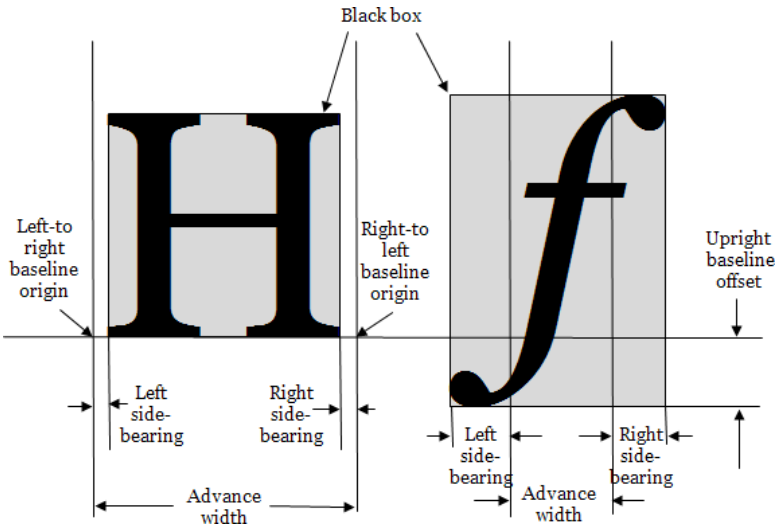
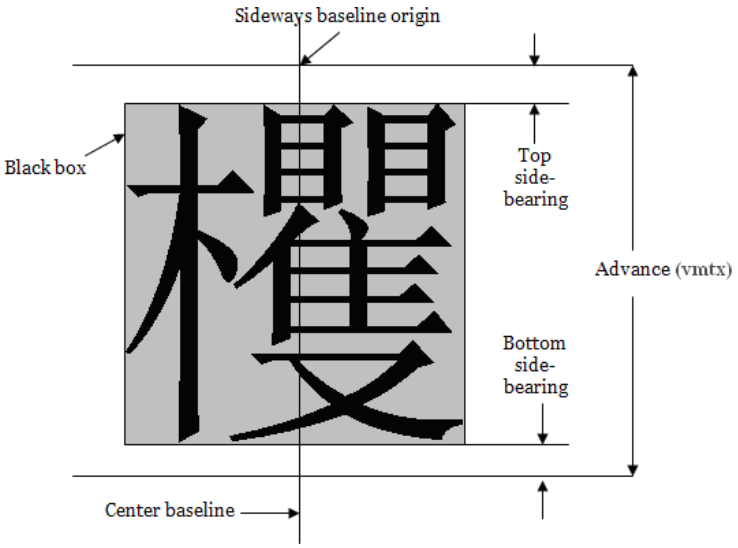


Figure 12–3. Sideways (usually vertical) glyph metrics



In general, glyphs within a font are either base glyphs or combining marks that can be attached to base glyphs. Base glyphs usually have an advance width that is non-zero, and a 0,0 offset vector. Combining marks usually have a zero advance width. The offset vector can be used to adjust the position of a combining mark and, therefore, can have a non-0,0 value for combining marks.

The position of each glyph in the glyph run is controlled by the following values:

- *Origin*. Each glyph is assumed to be given a nominal origin. For the first glyph in the run, this is the origin of the run.
- *Advance Width*. The advance width for each glyph provides the origin of the next glyph relative to the origin of the current glyph. The advance vector is drawn in the direction of the run progression.
- *Glyph Offset (Base or Mark)*. The glyph offset vector adjusts the position of this glyph relative to its nominal origin. The orientation of the glyph offset vector is not affected by the value of the IsSideways attribute, but is affected by the value of the BidiLevel attribute.

12.1.2 Mapping Code Units to Glyphs

A Unicode scalar value in a UnicodeString attribute is typically represented by a single UTF-16 code unit and has a single corresponding glyph representation in the font. More complex mapping scenarios are common in non-Latin scripts: a single Unicode scalar value can map to two UTF-16 code units, multiple UTF-16 code units can map to a single glyph, single UTF-16 code units can map to multiple glyphs based on context, and multiple UTF-16 code units can map indivisibly to multiple glyphs. In these cases, the clusters of UTF-16 code units are mapped using a cluster map.

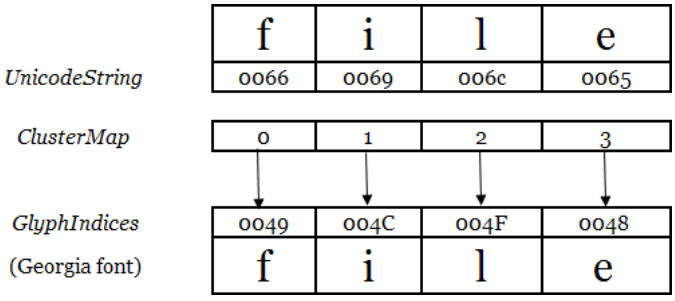
The cluster map contains one entry for each UTF-16 code unit in the UnicodeString attribute. Each entry specifies the offset of the first glyph that represents the cluster of UTF-16 code units.

12.1.2.1 One-to-One Mappings

When each UTF-16 code unit is represented by exactly one glyph, the cluster map entries are 0, 1, 2, and so on.

Example 12–1. One-to-one cluster map

Each character in the word "file" is represented by a single glyph.



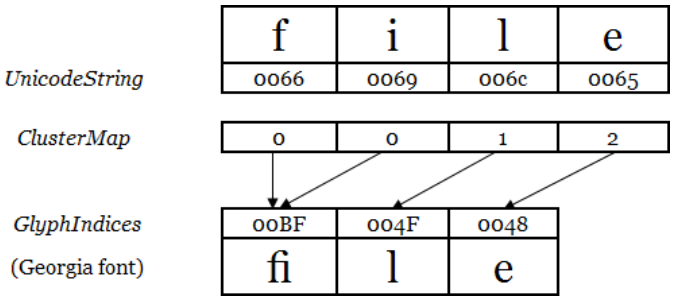
end example]

12.1.2.2 Many-to-One Mappings

When two or more UTF-16 code units map to a single glyph, the entries for those UTF-16 code units specify the offset of that glyph in the glyph index buffer.

Example 12–2. Many-to-one cluster map

In the following mapping, the *f* and *i* characters are replaced by a ligature.



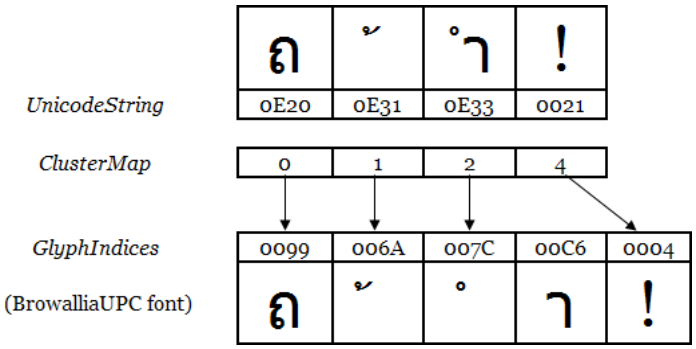
end example]

12.1.2.3 One-to-Many Mappings

When one UTF-16 code unit maps to two or more glyphs, the value in the cluster map for that UTF-16 code unit references the first glyph in the Indices attribute that represents that UTF-16 code unit.

Example 12–3. One-to-many cluster map

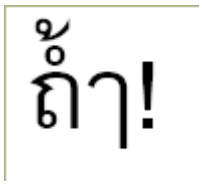
The Thai *Sara Am* character contains a part that sits on top of the previous base character (the ring), and a part that sits to the right of the base character (the hook). When Thai text is micro-justified, the hook is spaced apart from the base character, while the ring remains on top of the base character. Many fonts encode the ring and the hook as separate glyphs.



The markup appears as follows:

```
<Glyphs
   FontUri="../Resources/Fonts/browau.ttf"
   UnicodeString="&#xe20;&#xe3149;&#xe33;&#x21;"
   Indices="153;106,,,16;(1:2)124;198;4"
   OriginX="10" OriginY="60"
   FontRenderingEmSize="70"
   Fill="#000000"/>
```

The markup above is rendered as follows:



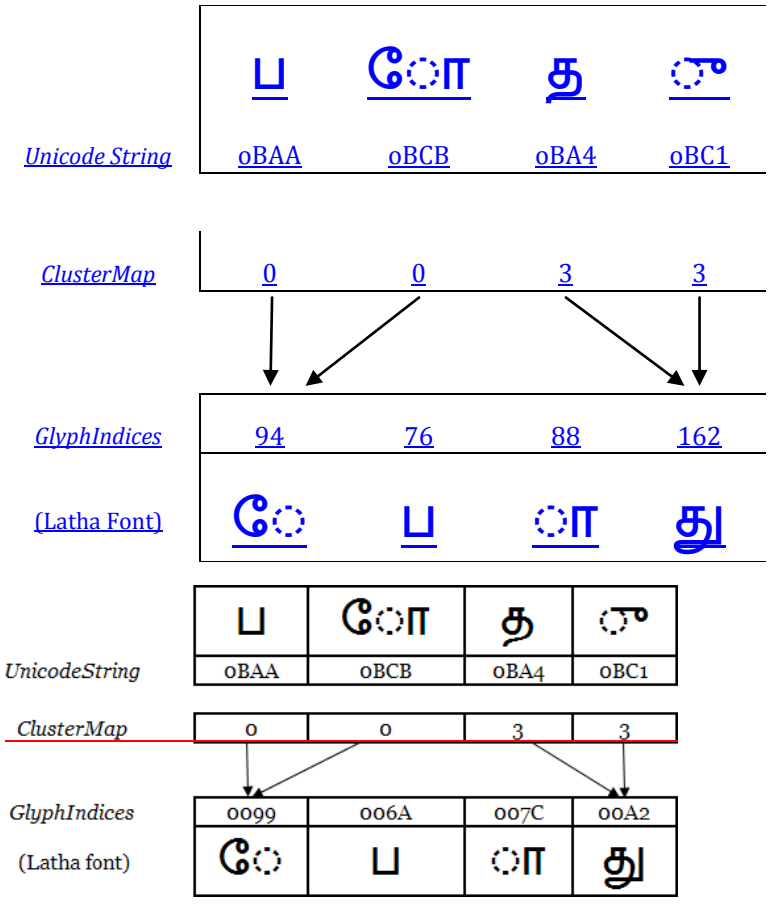
end example]

12.1.2.4 Many-to-Many Mappings

In some fonts, an indivisible group of UTF-16 code units for a character maps to more than one glyph. This is common in fonts that support Indic scripts. When an indivisible group of UTF-16 code units maps to one or more glyphs, the value in the cluster map for each of the UTF-16 code units references the first glyph in the Indices attribute representing that codepoint.

Example 12–4. Many-to-many cluster map

The following mapping shows the Unicode and glyph representations of a Tamil word that has two glyph clusters. Each cluster has a base character and a combining mark. The first pair of UTF-16 code units generates three glyphs because the combining mark splits both sides of the base character. The second pair of UTF-16 code units is represented by a single glyph that incorporates the effect of the combining mark.



The markup appears as follows:

```
<Glyphs
    FontUri="../Resources/Fonts/latha.ttf"
    UnicodeString="&#xbaa;&#xbcb;&#xba4;&#xbc1;"
    Indices="(2:3)94;76;88;(2:1)162"
    OriginX="10" OriginY="120"
    FontRenderingEmSize="40"
    Fill="#000000"/>
```

The markup above is rendered as follows:

end example]

12.1.3 Indices Attribute

The <Glyphs> element MAY have an Indices attribute [M2.72]. The glyph specifications within the Indices attribute are OPTIONAL [M2.72]. The GlyphIndex portion of the Indices attribute MAY be used to specify a series of glyphs, complex character-to-glyph cluster mappings, or a combination of both [M2.72]. The Indices attribute MAY also include glyph placement information [M2.72].

Within the Indices attribute, each glyph specification is separated by a semicolon. The Indices attribute MUST adhere to the glyph specification syntax as follows [M2.72]:

```
GlyphIndices   = *1GlyphMapping *( ";" *1GlyphMapping )
GlyphMapping   = *1([ClusterMapping] GlyphIndex) [GlyphMetrics]
ClusterMapping = "(" ClusterCodeUnitCount [":" ClusterGlyphCount] ")"
ClusterCodeUnitCount = 1*DIGIT
ClusterGlyphCount    = 1*DIGIT
GlyphIndex     = *DIGIT
GlyphMetrics   = "," *1AdvanceWidth ["," *1uOffset ["," vOffset]]
AdvanceWidth   = ["+"] RealNum
uOffset        = ["+" | "-"] RealNum
vOffset        = ["+" | "-"] RealNum
RealNum        = ((1*DIGIT ["." 1*DIGIT]) | ("." 1*DIGIT)) [Exponent]
Exponent       = *1( ("E"|"e") ("+"|"-") 1*DIGIT )
```

The sum of the code unit counts for all the GlyphMapping entries in the Indices attribute MUST NOT exceed the number of UTF-16 code units in the UnicodeString attribute if the UnicodeString attribute is specified and does not contain an empty value (“” or “{}”). If a ClusterMapping is not specified within a GlyphMapping entry, the code unit count is 1 [M5.4]. If the Indices attribute specifies a GlyphIndex that does not exist in the font, the consumer MUST generate an error [M5.24M5.4]. If the Indices attribute is specified, the values provided MUST be used in preference to values determined from the UnicodeString attribute alone [M5.23].

Table 12–3. Glyph specifications

Name	Description
GlyphIndex	Index of the glyph (16-bit) in the physical font. The entry MAY be empty [M2.72], in which case the glyph index is determined by looking up the UTF-16 code unit in the font character map table. If there is not a one-to-one mapping between code units and the glyph indices, this entry MUST be specified [M5.5]. In cases where character-to-glyph mappings are not one-to-one, a cluster mapping specification precedes the glyph index (further described below).
AdvanceWidth	Advance width indicating placement for the subsequent glyph, relative to the origin of the current glyph. Measured in direction of advance as defined by the IsSideways and BidiLevel attributes. Base glyphs generally have a non-zero advance width and combining glyphs have a zero advance width. Advance width is measured in hundredths of the font em size. The default value is defined in the horizontal metrics font table (hmtx) if the IsSideways attribute is specified as false or the vertical metrics font table (vmtx) if the IsSideways attribute is specified as true. Advance width is a real number with units specified in hundredths of

	an em. So that rounding errors do not accumulate, the advance MUST be calculated as the exact unrounded origin of the subsequent glyph minus the sum of the calculated (that is, rounded) advance widths of the preceding glyphs [M5.6]. The advance MUST be 0 or greater [M2.72]. The right-to-left writing direction can be specified using the BidiLevel attribute.
uOffset, vOffset	Offset in the effective coordinate space relative to glyph origin to move this glyph (*x* offset for uOffset and *–y* offset for vOffset. The sign of vOffset is reversed from the direction of the y axis. A positive vOffset value shifts the glyph by a negative y offset and vice versa.). Used to attach marks to base characters. The value is added to the nominal glyph origin calculated using the advance width to generate the actual origin for the glyph. The setting of the IsSideways attribute does not change the interpretation of uOffset and vOffset. Measured in hundredths of the font em size. The default offset values are 0.0,0.0. uOffset and vOffset are real numbers. Base glyphs generally have a glyph offset of 0.0,0.0. Combining glyphs generally have an offset that places them correctly on top of the nearest preceding base glyph. For left-to-right text, a positive uOffset value points to the right; for right-to-left text, a positive uOffset value points to the left.

Example 12–5. Using indices to specify advance width

The following Indices attribute specifies that the seventh glyph in the Unicode string has an advance width of 40:

```
Indices = ";;;;;;,40"
```

end example]

12.1.3.1 Specifying Character-to-Glyph Mappings

A cluster map specification MAY precede the glyph specification for the first glyph of the cluster [M2.72].

Empty Indices attribute values indicate that the corresponding UTF-16 code unit within the Unicode string has a one-to-one relationship with the glyph index as specified by the character mapping table within the font.

Cluster maps that specify 0:n or n:0 mappings are invalid.

See the glyph specification syntax above for details of how to specify cluster maps.

Table 12–4. Portions of the cluster specification

Name	Description
ClusterCodeUnitCount	Number of UTF-16 code units that combine to form this cluster. One or more code units can be specified. Default value is 1.
ClusterGlyphCount	Number of glyph indices that combine to form this cluster. One or more indices can be specified. Default value is 1.

Example 12–6. Using the Indices attribute to specify glyph replacement for a cluster

The following Indices attribute specifies that the sixth and seventh UTF-16 code units in the Unicode string should be replaced by a single glyph having an index of 191:

```
Indices = ";;;;;(2:1)191"
```

end example]

12.1.4 UnicodeString Attribute

The UnicodeString attribute holds the array of Unicode scalar values that are represented by the current <Glyphs> element. Specifying a Unicode string is RECOMMENDED, as it supports searching, selection, and accessibility [S5.5]. If the Unicode string contains Unicode scalar values that require two UTF-16 code units, a cluster map with a many-to-one or many-to-many mapping MUST be specified for the values [M5.5].

The standard XML escaping mechanisms are used to specify XML-reserved characters. An additional mechanism MUST be used to escape a UnicodeString attribute value that begins with an open brace ("{") [M5.7].

In order to use an open brace at the beginning of the Unicode string, it MUST be escaped with a prefix of "{}" [M5.7]. If the UnicodeString attribute value starts with "{}", consumers MUST ignore those first two characters in processing the Unicode string and in calculating index positions for the characters of the Unicode string [M5.7].

If the UnicodeString attribute specifies an empty string ("" or "{}"), and the Indices attribute is missing or is also empty, it MUST be treated as an error [M5.2]. If the UnicodeString attribute contains a Unicode code unit that cannot be mapped to a glyph index via a cmap table in the font and there is no corresponding GlyphIndex entry in the Indices attribute, the consumer MUST display the .notdef glyph [M5.9].

Producers MAY include Unicode control marks in the Unicode string [O5.1]. Such marks include control codes, layout controls, invisible operators, deprecated format characters, variation selectors, non-characters, and specials, according to their definition within the Unicode specification. If producers include control marks in the Unicode string, they SHOULD include an Indices attribute to specify glyph indices and/or character-to-glyph mapping information for the control marks [S5.2]. In the absence of such information, consumers MUST treat Unicode control marks like ordinary characters and render the glyphs to which the Unicode control marks are mapped in the CMAP table [M5.10]. The resulting glyphs might produce an inappropriate rendering of the original Unicode string.

Producers MAY choose to generate UnicodeString attribute values that are not normalized by any Unicode-defined algorithm [O5.2]. Because advance-widths, glyph indices, and caret-stops are associated with the generated Unicode string, consumers MUST NOT normalize the UnicodeString attribute value to produce an internal representation [M5.11]. See §9.1.7.5 for details and exceptions.

12.1.5 StyleSimulations Attribute

Synthetic style simulations can be applied to the shape of the glyphs by using the StyleSimulations attribute. Style simulations can be applied in addition to the designed style of a font. The default value for the StyleSimulations attribute is None, in which case the shapes of glyphs are not modified from their original design.

When the StyleSimulations value is specified as BoldSimulation, synthetic emboldening is applied by geometrically widening the strokes of glyphs by 1% of the em size, so that the centers of strokes remain at the same position. This leaves the baseline origin unmodified. The black box grows 1% all around for a total of 2% horizontal and 2% vertical. As a result, the character height and the advance width of each glyph are increased by 2% of the em size. Producers MUST lay out algorithmically emboldened glyphs using advance widths that are 2% of the em size larger than when not algorithmically emboldened [M5.12].

Consumers MUST implement the effect of algorithmic emboldening such that the black box of the glyph grows by 2% of the em size [M5.13]. When advance widths are omitted from the markup and the glyphs are algorithmically emboldened, the advance widths obtained from the horizontal metrics font table (if IsSideways is false) or the vertical metrics font table (if IsSideways is true) of the font MUST be increased by 2% of the em size [M5.13].

When StyleSimulations is specified as ItalicSimulation, synthetic italicizing is applied to glyphs with an IsSideways value of false by skewing the top edge of the alignment box of the character by 20° to the right, relative to the baseline of the character. Glyphs with an IsSideways value of true are italicized by skewing the right edge of the alignment box of the character by 20° down, relative to the baseline origin of the glyph. The character height and advance width are not modified. Producers MUST lay out algorithmically italicized glyphs using exactly the same advance widths as when not algorithmically italicized [M5.14].

When StyleSimulations is specified as BoldItalicSimulation, both BoldSimulation and ItalicSimulation are applied, in order.

12.1.6 IsSideways Attribute

Glyphs for text in vertical writing systems are normally represented by rotating the coordinate system and using the IsSideways attribute. <Glyphs> elements with the IsSideways attribute set to true will be rotated 90° counter-clockwise and placed so that the sideways baseline origin is coincident with the nominal origin of the character (within the glyph-local coordinate space), as modified by the offset vector in the Indices attribute. The advance vector places the nominal origin of the next character a distance along the direction of progression of the run. The direction of the advance vector is unaffected by IsSideways, however the method by which the size of the advance vector is chosen is different.

[*Example*: To represent a run of characters top to bottom on a page, a render transform can be used to rotate the <Glyphs> coordinate system 90° clockwise. OriginX and OriginY can be used to specify a position at the top of the column of text. Text from a vertical writing system can then be written using <Glyphs> elements with the IsSideways attribute set to true. The individual glyphs appear in the normal orientation because the rotation effected by the IsSideways attribute undoes the effect of the render transform. *end example*]

Text from horizontal writing systems can be included in the column by using <Glyphs> elements without specifying IsSideways, or using a value of false for it. The rotated coordinate system makes them appear top to bottom on the page, but with the glyphs rotated to the right.

If alternate vertical character representations are available in the font, the producer SHOULD use those and provide their glyph indices in the Indices attribute [S5.3].

12.1.6.1 Calculating Sideways Text Origin and Advance Width

The formulas below describe the method used to calculate each glyph's nominal origin, which is used for positioning the glyphs on the fixed page and for calculating the default advance width for each glyph.

The origin is the top center of the unturned glyph. The *x* origin of the unturned glyph is calculated to be exactly one-half the advance width of the glyph, as specified in the horizontal metrics table of the font. This formula is expressed as follows (in pseudocode):

```
TopOriginX = hmtx.advanceWidth[GlyphIndex] / 2
```

If the font is a CFF OpenType font, the *y* origin of the unturned glyph is determined from the vertical origin (vorg) table for the font, which can be specified for a particular glyph index but falls back to the default vertical origin if the glyph index is not present in the vertical origin table. This formula is expressed as follows (in pseudocode):

```
TopOriginY = vorg.vertOriginY[glyphIndex]
```

or:

```
TopOriginY = vorg.defaultVertOriginY
```

If the vertical origin table is not present, the glyph data (glyf) and vertical metrics (vmtx) font tables are consulted. The glyph bounding box is retrieved from the glyph data table and added to the top side-bearing for the glyph, specified in the vertical metrics table. This formula is expressed as follows (in pseudocode):

```
TopOriginY = glyf.yMax[glyphIndex] + vmtx.topSideBearing[glyphIndex]
```

[*Note*: CFF fonts do not contain the glyf.yMax information; instead the yMax for each glyph is computed by calculating the top of the glyph's bounding box from the CFF charstring data. *end note*]

If the vertical metrics font table does not exist but the Windows-specific metrics (OS/2) table does exist, the latter table is consulted and the sTypoAscender value is used. This formula is expressed as follows (in pseudocode):

```
TopOriginY = os/2.sTypoAscender
Descender = abs(os/2.typoDescender)
```

In all other circumstances, the Ascender value from the horizontal header (hhea) table is used. This formula is expressed as follows (in pseudocode):

```
TopOriginY = hhea.Ascender
Descender = abs(hhea.Descender)
```

Finally, the advance width for sideways text is computed as follows (in pseudocode), unless specifically overridden by the Indices attribute:

```
AdvanceWidth = TopOriginY + Descender
```

12.1.6.2 IsSideways and BidiLevel Effects on Glyph Positioning

Right-to-left text (BidiLevel attribute value of 1) changes the direction of the AdvanceWidth and uOffset (horizontal offset) values of the Indices attribute, as well as the position of the glyph origin. Vertical text (IsSideways attribute set to true) changes the position of the glyph origin.

Producers MUST NOT specify text that is both right-to-left (BidiLevel attribute value of 1) and vertical (IsSideways attribute set to true) [M5.15].

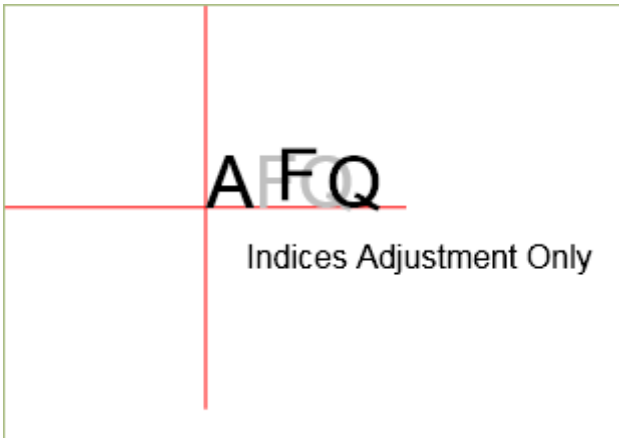
1 *Table 12–5. IsSideways and BidiLevel effects on origin placement*

IsSideways	BidiLevel	Glyph origin	Direction of advance width and positive uOffset
Horizontal (false)	Left-to-right (0)	Left end of horizontal advance vector along Latin baseline	To the right
Horizontal (false)	Right-to-left (1)	Right end of horizontal advance vector along Latin baseline	To the left
Vertical (true)	Left-to-right (0)	Top end of vertical advance vector through the glyph centerline	To the right
Vertical (true)	Right-to-left (1)	*Invalid combination*	

Example 12–7. Text with positive uOffset and vOffset Indices values

In this example, the position of the glyphs is shown relative to the origin shown at the crossed lines centered at 100,100. The text in gray shows where this text would be rendered without modification of the uOffset and vOffset value of the Indices attributes.

```
<Glyphs Fill="#000000" FontRenderingEmSize="48"
   OriginX="100" OriginY="100"
   UnicodeString="AFQ"
   Indices=";,100,30,10;"
   FontUri="../Resources/Fonts/Arial.ttf" />
```

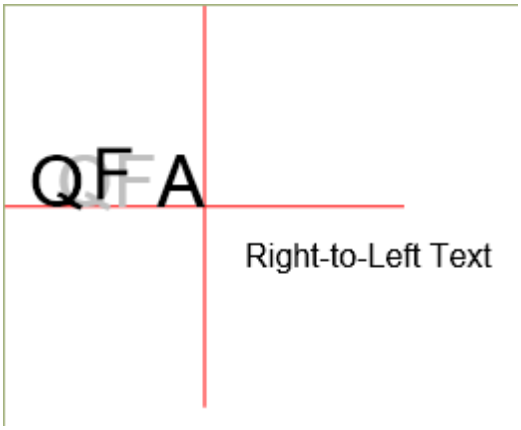


end example]

Example 12–8. Right-to-left text (odd BidiLevel)

The markup for this example matches the previous example, except the BidiLevel attribute is set to 1. Note the change in the origin, and the reversal of the glyph advance direction.

```
<Glyphs Fill="#000000" FontRenderingEmSize="48"
   OriginX="100" OriginY="100"
   UnicodeString="AFQ"
   Indices=";,100,30,10;"
   BidiLevel="1"
   FontUri="../Resources/Fonts/Arial.ttf" />
```

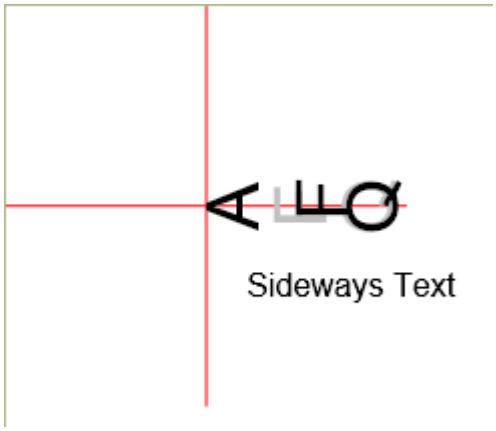


end example]

Example 12–9. Sideways text (IsSideways set to true)

This example shows the IsSideways attribute set to true. The BidiLevel ~~must~~ MUST be even when the IsSideways attribute is set to true [M5.15]. Note that the origin has changed to be the top-center of the first glyph, with each glyph rotated 90° counter-clockwise. The interpretation of the advance direction and uOffset and vOffset values in the Indices attribute are otherwise unchanged.

```
<Glyphs Fill="#000000" FontRenderingEmSize="48"
    OriginX="100" OriginY="100"
    UnicodeString="AFQ"
    Indices=";,100,30,10;"
    IsSideways="true"
    FontUri="../Resources/Fonts/Arial.ttf" />
```

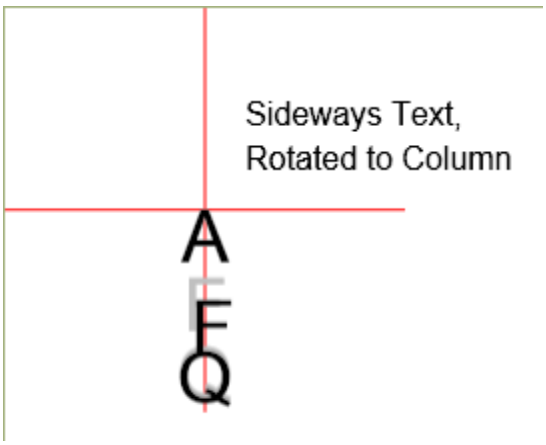


end example]

Example 12–10. Vertical text

The markup for this example matches the previous example, with the addition of a render transformation to rotate and position the element as vertical text. For more information on render transformations, see §14.4.

```
<Glyphs Fill="#000000" FontRenderingEmSize="48"
    OriginX="100" OriginY="100"
    UnicodeString="AFQ"
    Indices=";,100,30,10;"
    IsSideways="true"
    FontUri="../Resources/Fonts/Arial.ttf"
    RenderTransform="0,1,-1,0,200,0" />
```



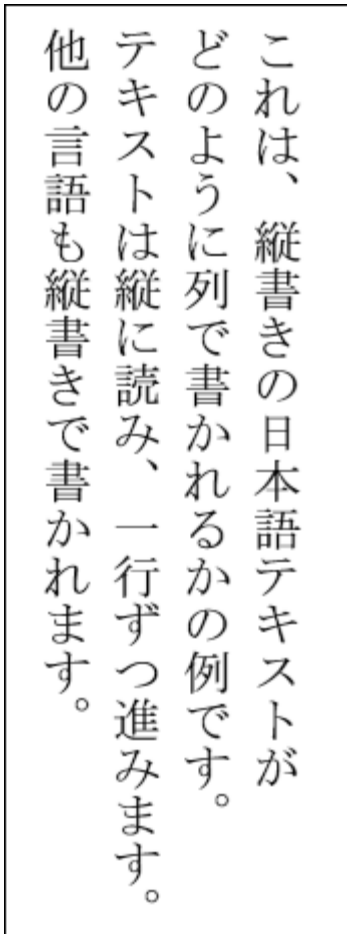
end example]

Example 12–11. Japanese vertical text

This example demonstrates a real-world usage of vertical text. Japanese text is shown below where the text is read down each column, from right to left across the page. The IsSideways attribute is set to true, thus rotating the each glyph 90° counter-clockwise. Then, the RenderTransform attribute (see §14.4) rotates the overall block of text 90° clockwise to achieve the final result of columns of text.

```
<Glyphs Fill="#000000" FontRenderingEmSize="24" OriginX="10" OriginY="10"
   UnicodeString="これは、縦書きの日本語テキストが"
   FontUri="../Resources/Fonts/msmincho.ttf" IsSideways="true"
   RenderTransform="0,1,-1,0,145,0"/>
<Glyphs Fill="#000000" FontRenderingEmSize="24" OriginX="10" OriginY="45"
   UnicodeString="どのように列で書かれるかの例です。"
   FontUri="../Resources/Fonts/msmincho.ttf" IsSideways="true"
   RenderTransform="0,1,-1,0,145,0"/>
<Glyphs Fill="#000000" FontRenderingEmSize="24" OriginX="10" OriginY="80"
   UnicodeString="テキストは縦に読み、一行ずつ進みます。"
   FontUri="../Resources/Fonts/msmincho.ttf" IsSideways="true"
   RenderTransform="0,1,-1,0,145,0"/>
<Glyphs Fill="#000000" FontRenderingEmSize="24" OriginX="10"
   OriginY="115" UnicodeString="他の言語も縦書きで書かれます。"
   FontUri="../Resources/Fonts/msmincho.ttf" IsSideways="true"
   RenderTransform="0,1,-1,0,145,0"/>
```

This markup is rendered as follows:



end example]

12.1.7 DeviceFontName Attribute

Printer device fonts are specified by the DeviceFontName attribute. Device manufacturers define the values for this attribute. Producers SHOULD NOT produce markup that will result in different rendering between consumers using the embedded font to render and consumers using the device font to render [S5.4].

Consumers that understand the device font name MAY ignore the embedded font and use the device-resident version [O5.3]. By definition, a consumer "understands" a printer device font if it can unambiguously correlate the device font name to a set of font metrics resident on the device. If a consumer does not understand the specified device font name, it MUST render the embedded version of the font [M5.16].

When rendering a printer device font, consumers MUST use the UnicodeString attribute and ignore the glyph index components of the Indices attribute [M5.17]. The consumer MUST still honor the advance width and x,y offset values present in the Indices attribute [M5.18].

For producers, a <Glyphs> element with a specified device font name MUST have exactly one Indices glyph per code unit in the UnicodeString attribute. Its Indices attribute MUST NOT include any cluster specifications. If the Indices attribute includes a cluster mapping, the consumer MUST NOT use the device font and MUST render the embedded version of the font [M5.19].

This means that a device font cannot be used for characters outside the basic multilingual plane (BMP).

If a device font name is specified, each of the <Glyphs> element's Indices glyphs MUST include a specified advance width and MUST include specified x and y offset values if they are non-zero [M5.20].

12.1.8 xml:lang Attribute

XPS Document consumers might need to override the default language for a specific run of glyphs, particularly in multilingual documents. The language defaults to the value specified for the xml:lang attribute of the <FixedPage> element but MAY be overridden by an xml:lang attribute on a <Glyphs> element [M2.72]. For larger blocks of text, the producer MAY specify the xml:lang attribute on the <Canvas> element [M2.72].

The language specified does not affect rendering of <Glyphs> elements, but it can be used by consumers for searching or selecting text. For more information, see §9.3.5.

12.1.9 CaretStops Attribute

The CaretStops attribute contains an array of Boolean bit-flags, which is represented as a string of hexadecimal characters. The flags indicate whether it is legal to place the caret before the corresponding UTF-16 code unit in the UnicodeString attribute. ("Before" refers to a *logical* placement, not a *physical* placement.) [*Example*: If the flag is set in right-to-left text, the caret can be placed before (to the right of) that UTF-16 code unit. *end example*] The CaretStops attribute includes a final flag for placement of the caret following the final UTF-16 code unit in the Unicode string.

Each hexadecimal character in the CaretStops value represents the flags for four UTF-16 code units in the Unicode string, with the highest-order bit representing the first UTF-16 code unit. Any unused bits in the last UTF-16 code unit must be 0.

If the CaretStops attribute is omitted, it is legal to place the caret before any of the UTF-16 code units in the Unicode string. Therefore, omitting the CaretStops attribute is equivalent to specifying a string that has all the bits set to 1. If there are insufficient flags in the CaretStops string to correspond to all the UTF-16 code units in the Unicode string, all remaining UTF-16 code units in the Unicode string MUST be considered valid caret stops [M5.22].

Example 12–12. Using the CaretStops attribute to determine a valid caret stop position

Given the following attributes, the *m* in "example" is not a valid caret stop position:

```
UnicodeString = "This is an example string of text."
CaretStops = "fffd"
```

end example]

12.1.10 Optimizing Glyph Markup

Markup details such as glyph indices and advance widths can be omitted from the markup under the circumstances described below. The following options allow optimization of commonly used simple scripts.

12.1.10.1 Optimizing Glyph Indices Markup

Glyph indices MAY be omitted from markup where *all* of the following are true [O5.4]:

- There is a one-to-one mapping between the positions of Unicode scalar values in the UnicodeString attribute and the positions of glyphs in the glyph string.
- The glyph index is the value in selected character mapping table of the font.

12.1.10.2 Optimizing Glyph Position Markup

Glyph advance width MAY be omitted from the markup in the following cases [O5.5]:

- For glyphs that have not been algorithmically emboldened, the desired advance width is the value listed in the horizontal metrics font table (if the IsSideways attribute value is false) or as calculated in §12.1.6.1 (if the IsSideways attribute value is true).
- For algorithmically emboldened glyphs, the desired advance width is exactly 2% larger than the values in the horizontal metrics font table (if the IsSideways attribute value is false) or as calculated in §12.1.6.1 (if the IsSideways attribute value is true).

Glyph horizontal offset MAY be omitted from the markup when the offset is 0.0, and Glyph vertical offset MAY be omitted from the markup when the offset is 0.0 [O5.6]. This is almost always true for base characters, and commonly true for combining marks in simple scripts. However, this is often false for combining marks in complex scripts such as Arabic and Indic.

12.1.11 Glyph Markup Examples

Example 12–13. Basic italic font

```
<Canvas>
   <Glyphs
      FontUri="../Resources/Fonts/Timesi.ttf"
      FontRenderingEmSize="20"
      OriginX="35"
      OriginY="35"
      UnicodeString="Basic italic font..."
      Fill="#009900" />
</Canvas>
```

This text is rendered as follows:

end example]

Example 12–14. Italic font using StyleSimulations attribute

```
<Canvas>
   <Glyphs
      FontUri="../Resources/Fonts/Times.ttf"
      FontRenderingEmSize="20"
      StyleSimulations="ItalicSimulation"
      OriginX="35"
      OriginY="35"
      UnicodeString="Simulated italic font..."
      Fill="#009900" />
</Canvas>
```

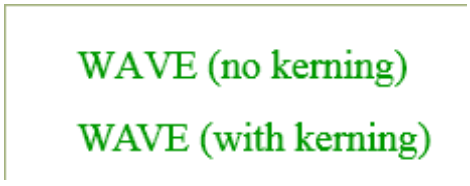
This text is rendered as follows:

end example]

Example 12–15. Kerning

```
    <Canvas>

       <!-- "WAVE" without kerning -->

       <Glyphs
          OriginX="35"
          OriginY="35"
          UnicodeString="WAVE (no kerning)"
          FontUri="../Resources/Fonts/Times.ttf"
          FontRenderingEmSize="20"
          Fill="#009900" />

       <!-- "WAVE" with kerning -->

       <Glyphs
          OriginX="35"
          OriginY="70"
          UnicodeString="WAVE (with kerning)"
          Indices=",88;,59"
          FontUri="../Resources/Fonts/Times.ttf"
          FontRenderingEmSize="20"
          Fill="#009900" />

    </Canvas>
```

This text is rendered as follows:



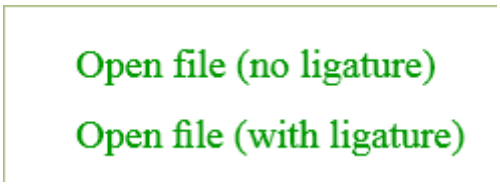
end example]

Example 12–16. Ligatures

```
    <Canvas>

       <!-- "Open file" without "fi" ligature -->

       <Glyphs
          OriginX="35"
          OriginY="35"
          UnicodeString="Open file (no ligature)"
          FontUri="../Resources/Fonts/Times.ttf"
          FontRenderingEmSize="20"
          Fill="#009900" />
```

```
        <!-- "Open file" with "fi" ligature -->

        <Glyphs
           OriginX="35"
           OriginY="70"
           UnicodeString="Open file (with ligature)"
           Indices=";;;;;(2:1)191"
           FontUri="../Resources/Fonts/Times.ttf"
           FontRenderingEmSize="20"
           Fill="#009900" />

    </Canvas>
```

This text is rendered as follows:



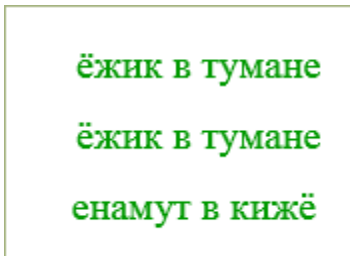
end example]

Example 12–17. Cluster maps

```
    <Canvas>

        <!-- "ёжик в тумане" using pre-composed "ё" -->

        <Glyphs
           OriginX="35"
           OriginY="35"
           xml:lang="ru-RU"
           UnicodeString="ёжик в тумане"
           FontUri="../Resources/Fonts/Times.ttf"
           FontRenderingEmSize="20"
           Fill="#009900" />

        <!-- "ёжик в тумане" using composition of "е" and diaeresis -->

        <Glyphs
           OriginX="35"
           OriginY="70"
           xml:lang="ru-RU"
           UnicodeString="ёжик в тумане"
           Indices="(1:2)72;142,0,-40"
           FontUri="../Resources/Fonts/Times.ttf"
           FontRenderingEmSize="20"
           Fill="#009900" />

        <!-- "ёжик в тумане" Forced rendering right-to-left showing
        combining mark in logical order -->

        <Glyphs
```

```
            OriginX="155"
            OriginY="105"
            BidiLevel="1"
            xml:lang="ru-RU"
            UnicodeString="ёжик в тумане"
            Indices="(1:2)72;142,0,-40"
            FontUri="../Resources/Fonts/Times.ttf"
            FontRenderingEmSize="20"
            Fill="#009900" />

    </Canvas>
```

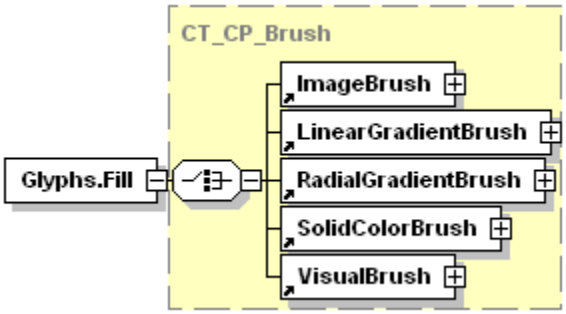
This text is rendered as follows:



end example]

12.2 <Glyphs.Fill> Element

element **Glyphs.Fill**

diagram	
annotation	Describes the brush used to fill the shape of the rendered glyphs.

The Fill property specifies the brush that fills a glyph. Any brush can be used.

17.1.1 Empty PrintTicket

An empty PrintTicket has the following form:

```
<psf:PrintTicket
xmlns:psf="http://schemas.microsoft.com/windows/2003/08/printing/printschemaframe
work" version="1"/>
```

It is RECOMMENDED that one empty PrintTicket be shared for all parts that attach an empty PrintTicket [S10.6].

17.1.2 Optimizing Interleaving Order

Producers MAY optimize the interleaving order of parts to help consumers avoid stalls during read-time streaming, and to allow consumers to manage their memory resources more efficiently [O10.2].

The optimization strategy is suggested by the consumer architecture. Therefore, interleaving optimization is typically implemented by a software component such as a driver or filter that is specific to (or aware of) the consumer architecture.

17.1.2.1 Single-Threaded Parsing Architectures

An optimal interleaving scheme for consumers with a single-threaded parsing model interleaves parts so that each part that is required to consume a single page (FixedPage, images, and fonts) is contained in the package in its entirety, prior to the FixedPage part being referenced from the FixedDocument part's markup.

Single-threaded parsing architectures typically require more run-time memory resources than multi-threaded parsing architectures because the context in which a resource is used is unknown at the time the resource is received. This requires deferred processing and additional buffering.

[*Note*: When interleaving entities containing XML markup, such as the DiscardControl part, the Content Types stream, and the FixedDocument part, there is no guarantee that XML element boundaries will align with piece boundaries in the physical package. This adds a complexity to single-threaded parsing architectures: the parser must be pre-emptable. Certain existing XML parser implementations might require a pre-tokenization step. *end note*]

Example 17–1. Optimized interleaving for a single-threaded parsing architecture

The following markup describes a sequence of two fixed documents, the first having two FixedPage parts and the second having one FixedPage part:

Part/Piece	Markup
Font1.ttf	`...binary font data...`
Other resources	`...resource data...`
Page1	`<FixedPage xmlns="http://schemas.microsoft.com /xps/2005/06" ...>`
	`<Glyphs FontURI="Font1.ttf"/>`
	`</FixedPage>`
Page1.rels	`<Relationships xmlns=`

	"http://schemas.openxmlformats.org/package/2006/relationships">
	<Relationship Type= "http://schemas.microsoft.com/xps/2005/06 /required-resource" Target="Font1.ttf"/>
	</Relationships>
FixedDocument1/[0].piece	<FixedDocument xmlns= "http://schemas.microsoft.com/xps/2005/06">
	<PageContent Source="Page1"/>
Sequence1/[0].piece	<FixedDocumentSequence xmlns= "http://schemas.microsoft.com/xps/2005/06">
	<DocumentReference Source="FixedDocument1"/>
_rels/.rels/[0].piece	<Relationships xmlns= "http://schemas.openxmlformats.org/package/2006/relationships">
	<Relationship Type="StartPart" Target="Sequence1"/>
Page2	<FixedPage xmlns= "http://schemas.microsoft.com/xps/2005/06" ...>...</FixedPage>
FixedDocument1/[1].last.piece	<PageContent Source="Page2"/>
	</FixedDocument>
Page3	<FixedPage xmlns= "http://schemas.microsoft.com/xps/2005/06" ...>...</FixedPage>
FixedDocument2	<FixedDocument xmlns= "http://schemas.microsoft.com/xps/2005/06">
	<PageContent Source="Page3"/>
	</FixedDocument>
Sequence1/[1].last.piece	<DocumentReference Source="FixedDocument2" />
	</FixedDocumentSequence>
_rels/.rels/[1].last.piece	</Relationships>

end example]

17.1.2.2 Multi-Threaded Parsing Architectures

An optimal interleaving scheme for consumers with a multi-threaded parsing model interleaves parts so that each resource part that is required to consume a single page (images and fonts) is contained in the package after the FixedPage part referencing it.

Multi-threaded parsing architectures typically require less run-time memory resources than single-threaded parsing architectures because the context in which resources appear is fully determined and, therefore, resources can be processed immediately.

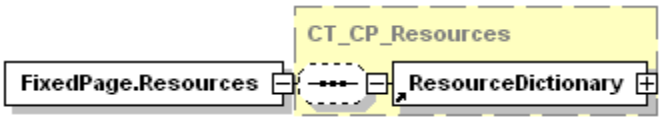
[*Note*: When interleaving entities containing XML markup, such as the DiscardControl part, the content type stream, and the FixedDocument part, there is no guarantee that XML element boundaries will align with piece boundaries in the physical package. A multi-threaded parsing architecture is naturally suited to address this problem. *end note*]

						element as a named, addressable point in the document for the purpose of hyperlinking.
annotation	Contains markup that describes the rendering of a single page of content.					

For more information, see§10.3.

19.17 FixedPage.Resources

element **FixedPage.Resources**

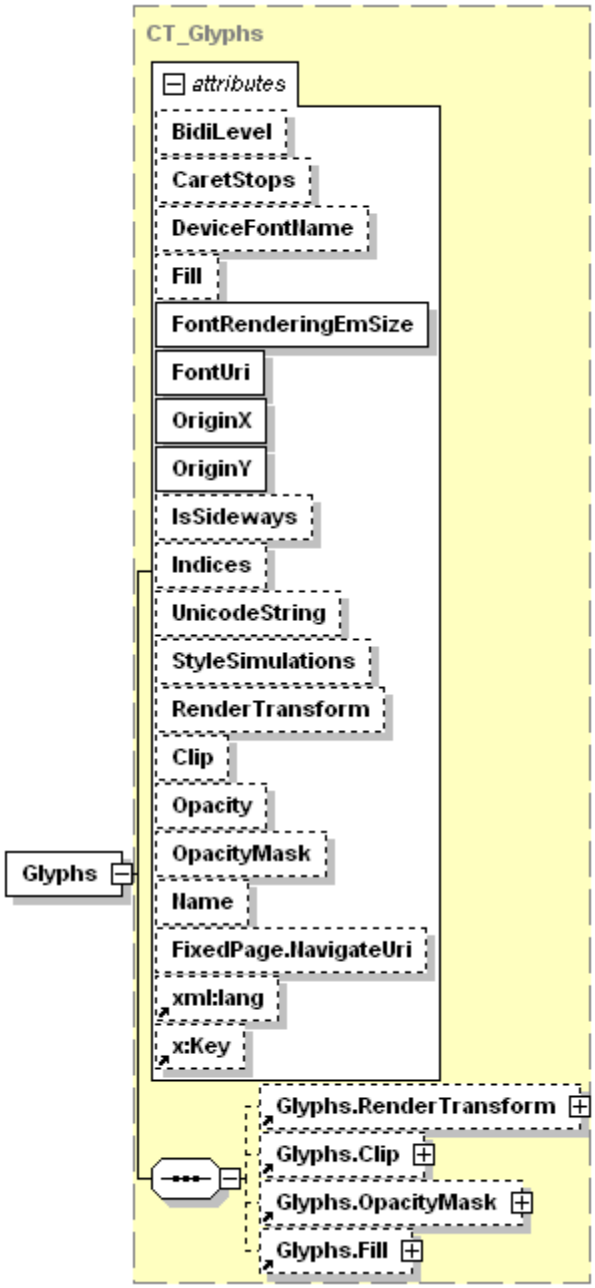
diagram	
annotation	Contains the resource dictionary for the <FixedPage> element.

For more information, see §14.2.

19.18 Glyphs

element **Glyphs**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
BidiLevel			0		Specifies the Unicode algorithm bidirectional nesting level. Even values imply left-to-right layout, odd values imply right-to-left layout. Right-to-left layout places the run origin at the right side of the first glyph, with positive advance widths (representing

						advances to the left) placing subsequent glyphs to the left of the previous glyph. Valid values range from 0 to 61, inclusive.
	CaretStops	ST_CaretStops				Identifies the positions within the sequence of Unicode characters at which a text-selection tool can place a text-editing caret. Potential caret-stop positions are identified by their indices into the UTF-16 code units represented by the UnicodeString attribute value. When this attribute is missing, the text in the UnicodeString attribute value MUST be interpreted as having a caret stop between every Unicode UTF-16 code unit and at the beginning and end of the text [M5.1]. The value SHOULD indicate that the caret cannot stop in front of most combining marks or in front of the second UTF-16 code unit of UTF-16 surrogate pairs [S5.1].
	DeviceFontName	ST_UnicodeString				Uniquely identifies a specific device font. The identifier is typically defined by a hardware vendor or font vendor.
	Fill	ST_RscRefColor				Describes the brush used to fill the shape of the rendered glyphs.
	FontRenderingEmSize	ST_GEZero	required			Specifies the font size in drawing surface units, expressed as a float in units of the effective coordinate space. A value of 0 results in no visible text.
	FontUri	xs:anyURI	required			The URI of the physical font from which all glyphs in the run are drawn. The URI MUST reference a font contained in the package [M2.1]. If the physical font referenced is a TrueType Collection (containing multiple font faces), the fragment portion of the URI is a 0-based index indicating which font face of the TrueType Collection should be used.

	OriginX	ST_Double	required			Specifies the x coordinate of the first glyph in the run, in units of the effective coordinate space. The glyph is placed so that the leading edge of its advance vector and its baseline intersect with the point defined by the OriginX and OriginY attributes.
	OriginY	ST_Double	required			Specifies the y coordinate of the first glyph in the run, in units of the effective coordinate space. The glyph is placed so that the leading edge of its advance vector and its baseline intersect with the point defined by the OriginX and OriginY attributes.
	IsSideways	ST_Boolean		false		Indicates that a glyph is turned on its side, with the origin being defined as the top center of the unturned glyph.
	Indices	ST_Indices				Specifies a series of glyph indices and their attributes used for rendering the glyph run. If the UnicodeString attribute specifies an empty string ("" or "{}") and the Indices attribute is not specified or is also empty, a consumer MUST generate an error [M5.2].
	UnicodeString	ST_UnicodeString				Contains the string of text rendered by the <Glyphs> element. The text is specified as Unicode code points.
	StyleSimulations	ST_StyleSimulations		None		Specifies a style simulation. Valid values are None, ItalicSimulation, BoldSimulation, and BoldItalicSimulation.
	RenderTransform	ST_RscRefMatrix				Establishes a new coordinate frame for the glyph run specified by the <Glyphs> element. The render transform affects clip, opacity mask, fill, x origin, y origin, the actual shape of individual glyphs, and the advance widths. The render transform also affects the font size and values specified in the

						Indices attribute.
	Clip	ST_RscRefAbbrGeomF				Limits the rendered region of the element. Only portions of the <Glyphs> element that fall within the clip region (even partially clipped characters) produce marks on the page.
	Opacity	ST_ZeroOne		1.0		Defines the uniform transparency of the glyph element. Values range from 0 (fully transparent) to 1 (fully opaque), inclusive. Values outside of this range are invalid.
	OpacityMask	ST_RscRef				Specifies a mask of alpha values that is applied to the glyphs in the same fashion as the Opacity attribute, but allowing different alpha values for different areas of the element.
	Name	ST_Name				Contains a string value that identifies the current element as a named, addressable point in the document for the purpose of hyperlinking.
	FixedPage.NavigateUri	xs:anyURI				Associates a hyperlink URI with the element. May be a relative reference or a URI that addresses a resource that is internal to or external to the package.
	xml:lang					Specifies the default language used for the current element. The language is specified according to RFC 3066.
	x:Key					Specifies a name for a resource in a resource dictionary. x:Key MUST be present when the current element is defined in a resource dictionary. x:Key MUST NOT be specified outside of a resource dictionary [M5.3].
annotation	Represents a run of text from a single font.					

1 For more information, see §12.1, §9.1.7, and §12.1.3.

H.2 Content Types

The content types in the tables below MUST NOT include parameters. A consumer MUST treat the presence of parameters on these content types as an error when the affected part is accessed [M12.7].

Table H–3. Package-wide content types

Description	Content type
Core Properties part	application/vnd.openxmlformats-package.core-properties+xml
Digital Signature Certificate part	application/vnd.openxmlformats-package.digital-signature-certificate
Digital Signature Origin part	application/vnd.openxmlformats-package.digital-signature-origin
Digital Signature XML Signature part	application/vnd.openxmlformats-package.digital-signature-xmlsignature+xml
Relationships part	application/vnd.openxmlformats-package.relationships+xml

Table H–4. XPS Document content types

Description	Content type
FixedDocument	application/vnd.ms-package.xps-fixeddocument+xml
FixedDocumentSequence	application/vnd.ms-package.xps-fixeddocumentsequence+xml
FixedPage	application/vnd.ms-package.xps-fixedpage+xml
DiscardControl	application/vnd.ms-package.xps-discard-control+xml
DocumentStructure	application/vnd.ms-package.xps-documentstructure+xml
Font	application/vnd.ms-opentype
ICC profile	application/vnd.ms-color.iccprofile
JPEG image	image/jpeg
Obfuscated font	application/vnd.ms-package.obfuscated-opentype
PNG image	image/png
PrintTicket	application/vnd.ms-printing.printticket+xml
Remote resource dictionary	application/vnd.ms-package.xps-resourcedictionary+xml
StoryFragments	application/vnd.ms-package.xps-storyfragments+xml
TIFF image	image/tiff
Thumbnail part	image/jpeg or image/png
Windows Media Photo image	image/vnd.ms-photo

H.3 Relationship Types

Table H–5. Package-wide relationship types

Description	Relationship type
Core Properties	http://schemas.openxmlformats.org/package/2006/relationships/metadata/core-properties
Digital Signature	http://schemas.openxmlformats.org/package/2006/relationships/digital-signature/signature
Digital Signature Certificate	http://schemas.openxmlformats.org/package/2006/relationships/digital-signature/certificate
Digital Signature Origin	http://schemas.openxmlformats.org/package/2006/relationships/digital-signature/origin
Thumbnail	http://schemas.openxmlformats.org/package/2006/relationships/metadata/thumbnail

Table H–6. XPS Document relationship types

Description	Relationship type
Digital Signature Definitions	http://schemas.microsoft.com/xps/2005/06/signature-definitions
DiscardControl	http://schemas.microsoft.com/xps/2005/06/discard-control
DocumentStructure	http://schemas.microsoft.com/xps/2005/06/documentstructure
PrintTicket	http://schemas.microsoft.com/xps/2005/06/printticket
Required Resource	http://schemas.microsoft.com/xps/2005/06/required-resource
Restricted Font	http://schemas.microsoft.com/xps/2005/06/restricted-font
StartPart	http://schemas.microsoft.com/xps/2005/06/fixedrepresentation
StoryFragments	http://schemas.microsoft.com/xps/2005/06/storyfragments

ID	Rule	Reference	Producer	Consumer
M2.3	An XPS Document MUST contain exactly one FixedDocumentSequence part per fixed payload.	9.1, 9.1.2	×	v
M2.4	An XPS Document MUST contain at least one FixedDocument part per fixed payload.	9.1	×	v
M2.5	An XPS Document MUST contain at least one FixedPage part per fixed payload.	9.1	×	v
M2.6	A <Glyphs> element in FixedPage markup MUST reference a Font part that exists in the XPS Document.	9.1	×	v
M2.7	An <ImageBrush> element in FixedPage markup MUST reference an Image part that exists in the XPS Document.	9.1	×	v
M2.8	If FixedPage markup references a Remote Resource Dictionary part, it MUST be included in the XPS Document	9.1	×	v
M2.9	This requirement was removed prior to Edition 1 of this specification.			
M2.10	Resources, which include fonts, images, color profiles, and remote resource dictionaries, that are referenced by URIs in FixedPage markup MUST use the Required Resource relationship from the FixedPage to the resource. If any resource references other resources, the indirectly required resource is also targeted by a Required Resource relationship from the FixedPage to the indirectly required resource.	9.1.1	×	v
M2.11	This requirement was removed prior to Edition 1 of this specification.			
M2.12	A Restricted Font relationship is REQUIRED for each print and preview font used, from the FixedDocument part to the preview and print Font part. When invoking editing functionality, a consumer that is also a producer MUST treat as an error any font with the print and preview licensing intent bit set for which no Restricted Font relationship has been added to the FixedDocument part. Printing and display-only consumers MUST consider an XPS Document valid, even if the producer failed to properly set the Restricted Font relationship.	9.1.1, 9.1.7.2, 9.1.7.4	×	v[E]
M2.13	Exactly one StartPart relationship is REQUIRED.	9.1.1	×	v
M2.14	The StartPart relationship MUST point from the package to the FixedDocumentSequence part that is the primary fixed payload root.	9.1~~9.1.2~~	×	v
M2.15	The order of <DocumentReference> elements in a FixedDocumentSequence part MUST be preserved by consumers that are also producers.	9.1.2		×[E]
M2.16	The order of <PageContent> elements in a FixedDocument MUST be preserved by consumers that are also producers.	9.1.3		×[E]
M2.17	JPEG image parts MUST contain images that conform to the	9.1.5.1	×	v[U]

ID	Rule	Reference	Producer	Consumer
	JPEG specification.			
M2.18	PNG image parts MUST contain images that conform to the PNG specification.	9.1.5.2	×	v^U
M2.19	The PNG ancillary chunk tRNS MUST be supported.	9.1.5.2		×
M2.20	The PNG ancillary chunk iCCP MUST be supported.	9.1.5.2		×
M2.21	The PNG ancillary chunk sRGB MUST be ignored.	9.1.5.2		×
M2.22	The PNG ancillary chunk cHRM MUST be ignored.	9.1.5.2		×
M2.23	The PNG ancillary chunk gAMA MUST be ignored.	9.1.5.2		×
M2.24	The PNG ancillary chunk sBIT MUST be ignored.	9.1.5.2		×
M2.25	TIFF image parts MUST contain images that conform to the TIFF specification	9.1.5.3	×	v^U
M2.26	XPS Document consumers MUST support baseline TIFF 6.0 with the tag values described in Table 9–5 for the specified TIFF image types, excepting the tags described in §9.1.5.3.	9.1.5.3		×
M2.27	If a TIFF file contains multiple image file directories (IFDs), consumers MUST use only the first IFD and ignore all others.	9.1.5.3		×
M2.28	XPS Document consumers MUST support TIFF images using CCITT bilevel encoding.	9.1.5.3		×
M2.29	XPS Document consumers MUST support CMYK TIFF images.	9.1.5.3		×
M2.30	XPS Document consumers MUST support TIFF images with associated alpha data. If the ExtraSamples tag is 1, the alpha is treated as pre-multiplied alpha. With an ExtraSamples tag of 2, the alpha is treated as non-pre-multiplied alpha.	9.1.5.3		×
M2.31	XPS Document consumers MUST support TIFF images using LZW compression.	9.1.5.3		×
M2.32	XPS Document consumers MUST support TIFF images using differencing predictors.	9.1.5.3		×
M2.33	XPS Document consumers MUST support TIFF images using JPEG compression (compression mode 6 only).	9.1.5.3		×
M2.34	XPS Document consumers MUST support TIFF images with an embedded ICC profile.	9.1.5.3		×
M2.35	Windows Media Photo image files MUST conform to the Windows Media Photo specification.	9.1.5.4	×	v^U
M2.36	Each FixedPage part MUST NOT have more than one thumbnail part attached.	9.1.6	×	v^T ∓
M2.37	Thumbnails MUST be either JPEG or PNG images	9.1.6	×	v^T
M2.38	If using a fragment in the FontURI attribute of the <Glyphs> element to indicate the font face to use from a TrueType Collection, the attribute value MUST be an integer between 0	9.1.7	×	v

ID	Rule	Reference	Producer	Consumer
	and *n*–1 inclusive, where *n* is the number of font faces in the TrueType Collection.			
M2.39	All fonts used in XPS Documents MUST adhere to the OpenType font format, which includes TrueTyep and CFF fonts. A subsetted font MUST still be a valid OpenType font file.	9.1.7, 9.1.7.1	×	v
M2.40	Producers MUST honor the licensing rights specified in OpenType fonts by following the embedding and obfuscation mechanisms described in this specification.	9.1.7.2	×	
M2.41	Consumers MUST be able to process XPS Documents using any combination of the embedding and obfuscation mechanisms described in this specification (even if produced in violation of the production requirements).	9.1.7.2		×
M2.42	For fonts with "Restricted license embedding" licensing intent, producers MUST NOT embed the font.	9.1.7.2	×	
M2.43	For fonts with "Print and preview embedding" licensing intent, consumers MUST NOT edit or modify any part of the XPS Document markup or hierarchical structure from the FixedDocument containing such a font downwards.	9.1.7.2, 9.1.7.4		×[E]
M2.44	For fonts with "Print and preview embedding" licensing intent, producers MUST perform embedded font obfuscation.	9.1.7.2	×	
M2.45	For fonts with "Print and preview embedding" licensing intent, consumers MUST NOT extract or permanently install the font.	9.1.7.2		×
M2.46	For fonts with "Editable embedding" licensing intent, producers MUST perform embedded font obfuscation.	9.1.7.2	×	
M2.47	For fonts with "Editable embedding" licensing intent, consumers MUST NOT extract or permanently install the font.	9.1.7.2		×
M2.48	For fonts with "No subsetting" licensing intent, producers MUST perform embedded font obfuscation.	9.1.7.2	×	
M2.49	For fonts with "No subsetting" licensing intent, producers MUST NOT subset the font.	9.1.7.2	×	
M2.50	For fonts with "No subsetting" licensing intent, consumers MUST NOT extract or permanently install the font.	9.1.7.2		×
M2.51	For fonts with "Bitmap embedding only" licensing intent, producers MUST perform embedded font obfuscation for bitmap characters only. If no bitmap characters are present in the font, the producer MUST NOT embed the font.	9.1.7.2	×	
M2.52	For fonts with "Bitmap embedding only" licensing intent, consumers MUST NOT extract or permanently install the font.	9.1.7.2		×
M2.53	Producers and consumers MUST perform font obfuscation and de-obfuscation according to the steps described in §9.1.7.3.	9.1.7.3	×	×
M2.54	The last segment of the part name for an obfuscated font MUST	9.1.7.3	×	v

ID	Rule	Reference	Producer	Consumer
	be the GUID generated during the font obfuscation process, with or without an extension.			
M2.55	When processing <Glyphs> elements, the consumer MUST select a cmap table from the OpenType font according to Table 1-8 in §9.1.7.5. All further processing for that font MUST use the selected cmap table.	9.1.7.5	×	×
M2.56	When processing <Glyphs> elements, if a WanSung, Big5, Prc, ShiftJis, or MacRoman cmap has been selected, the consumer MUST correctly map from Unicode codepoints in the UnicodeString attribute to the corresponding codepoints used by the cmap before looking up glyphs.	9.1.7.5		×
M2.57	When processing <Glyphs> elements that reference a cmap (3,0) encoding font, consumers MUST handle the case where the UnicodeString attribute contains character codes instead of PUA codepoints by computing the correct glyph index according to the general recommendations of the OpenType specification.	9.1.7.5		×
M2.58	Consumers MUST process all PrintTicket parts when an XPS Document is printed.	9.1.9, 9.1.9.3		×[P]
M2.59	A level-specific PrintTicket MUST contain only settings scoped to the current level and child levels. Job-level PrintTicket parts MUST contain only job-, document-, and page-scoped settings; document-level PrintTicket parts MUST contain only document-scoped and page-scoped settings; and page-level PrintTicket parts MUST contain only page-scoped settings. Print schema elements that interact between levels MUST be specified at the root of each level ticket. Each FixedDocumentSequence, FixedDocument, or FixedPage part MUST have no more than one attached PrintTicket.	9.1.9, 9.1.9.2, 9.1.9.3	×[U]	
M2.60	Consumers MUST process job-level, document-level and page-level settings of PrintTicket parts associated with FixedDocumentSequence parts.	9.1.9.2		×[P]
M2.61	Consumers MUST process document-level and page-level settings of PrintTicket parts associated with FixedDocument parts and MUST ignore job-level settings of PrintTicket parts associated with FixedDocument parts.	9.1.9.2		×[P]
M2.62	Consumers MUST process page-level settings of PrintTicket parts associated with FixedPage parts and MUST ignore job-level and document-level settings of PrintTicket parts associated with FixedPage parts.	9.1.9.2		×[P]
M2.63	When processing a PrintTicket, consumers MUST first remove all levels of PrintTicket content not applicable to the current element.	9.1.9.3		×[P]
M2.64	When processing a PrintTicket, consumers MUST second validate the PrintTicket according to the methods defined in the	9.1.9.3		×[P]

ID	Rule	Reference	Producer	Consumer
	specification.			
M2.74	Properties MUST NOT be set more than once, regardless of the syntax used to specify the value. In certain cases, they can be specified using either property attributes or property elements. Consumers MUST treat properties that are specified in both ways as an error.	9.3.3.2	×	v
M2.75	XPS Document markup MUST NOT use the xml:space attribute.	9.3.4	×	v
M2.76	The language of the contents of an XPS Document MUST be identified using the xml:lang attribute, the value of which is inherited by child and descendant elements. When the language of the contents is unknown and is required, the value "und" (undetermined) MUST be used.	9.3.5.1	×	
M2.77	Producers that generate a relationship MUST include the target part in the XPS Document for any of the following relationship types: DiscardControl, DocumentStructure, PrintTicket, Required Resource, Restricted Font, StartPart, StoryFragments, and Thumbnail. Consumers that access the target part of any relationship with one of these relationship types MUST generate an error if the part is not included in the XPS Document.	9.1.1	×[U]	v[U]
M2.78	Consumers MUST support JPEG images that contain the APP1 marker and interpret the EXIF color space correctly.	9.1.5.1~~9.1.5.2~~		×
M2.79	XPS Document consumers MUST support TIFF images that include the EXIF IFD (tag 34665) as described in the EXIF specification. The EXIF color space MUST be interpreted correctly.	9.1.5.3		×
M2.80	Each <DocumentReference> element in a FixedDocumentSequence part MUST reference a FixedDocument part by relative URI.	9.1.2	×	
M2.81	Each <PageContent> element in a FixedDocument part MUST reference a FixedPage part by relative URI.	9.1.3	×	
M2.82	<ImageBrush> and <Glyphs> elements MUST reference Image and Font parts by relative URI.	9.1.4	×	
M2.83	If the ExtraSamples tag value is 0, the associated alpha data in this channel MUST be ignored	9.1.5.3		×

ID	Rule	Reference	Producer	Consumer
	circulation, consumers SHOULD test as many different TIFF images as possible, correct common mistakes in TIFF images, and implement a reasonable recovery strategy when a problematic TIFF image is encountered.			
S2.12	It is RECOMMENDED that Windows Media Photo images end with the extension ".wdp".	9.1.5.4	×	
S2.13	It is RECOMMENDED that if thumbnails are used for pages, a thumbnail SHOULD be included for every page in the document.	9.1.6	×	
S2.14	Consumers SHOULD only process thumbnails associated via a package relationship from the package as a whole or via a relationship from a FixedPage part. Thumbnails attached to any other part SHOULD be ignored.	9.1.6		×
S2.15	Producers SHOULD use Unicode-encoded fonts.	9.1.7, 9.1.7.5	×	
S2.16	For fonts with "Installable embedding" licensing intent, producers SHOULD perform embedded font obfuscation.	9.1.7.2	×	
S2.17	For fonts with "Installable embedding" licensing intent, consumers SHOULD NOT extract or permanently install the font.	9.1.7.2		×
S2.18	For fonts with "Restricted license embedding" licensing intent, producers SHOULD generate a path filled with an image brush referencing an image of rendered characters and SHOULD include the actual text in the AutomationProperties.Name attribute of the <Path> element.	9.1.7.2	×	
S2.19	Although the licensing intent allows embedding of non-obfuscated fonts and installation of the font on a remote client system under certain conditions, this is NOT RECOMMENDED in XPS Documents. Instead, producers SHOULD always perform font obfuscation, and consumers SHOULD never extract or permanently install fonts.	9.1.7.3	×	×
S2.20	It is RECOMMENDED that the extension of an obfuscated Font part name be ".odttf" for TrueType fonts and ".odttc" for TrueType collections.	9.1.7.3	×	
S2.21	Producers SHOULD include only PrintTicket settings that support portability of the XPS Document.	9.1.9.1	×	
S2.22	Producers SHOULD only attach PrintTicket parts containing only document-level and page-level settings with FixedDocument parts.	9.1.9.2	×	
S2.23	Producers SHOULD only attach PrintTicket parts containing only page-level settings with FixedPage parts.	9.1.9.2 ~~9.1.9.3~~	×	
S2.24	The FixedDocumentSequence part SHOULD follow the part name recommendation "/<*FixedDocSeq*>.fdseq" where	9.2	×	

ID	Rule	Reference	Producer	Consumer
S2.35	If the referenced font part is a TrueType Collection, then if the fragment portion of the URI is not recognised as a valid integer, consumers SHOULD generate an error.	9.1.7		×

I.2.3 OPTIONAL Conformance Requirements

Table I–4. Parts and Relationships OPTIONAL conformance requirements

ID	Rule	Reference	Producer	Consumer
O2.1	Thumbnail parts MAY be included in an XPS Document	9.1	×	
O2.2	PrintTicket parts MAY be included in an XPS Document.	9.1	×	
O2.3	ICC Profile parts MAY be included in an XPS Document.	9.1	×	
O2.4	DocumentStructure parts MAY be included in an XPS Document.	9.1	×	
O2.5	StoryFragments parts MAY be included in an XPS Document.	9.1	×	
O2.6	SignatureDefinitions parts MAY be included in an XPS Document.	9.1	×	
O2.7	DiscardControl parts MAY be included in an XPS Document.	9.1	×	
O2.8	A Core Properties relationship MAY be included in an XPS Document, from the package to the Core Properties part.	9.1.1	×	
O2.9	A Digital Signatures Origin relationship MAY be included in an XPS Document, from the package to the Digital Signature Origin part.	9.1.1	×	
O2.10	Digital Signature relationships MAY be included in an XPS Document, from the Digital Signature Origin part to a Digital Signature XML Signature part.	9.1.1	×	
O2.11	Digital Signature Certificate relationships MAY be included in an XPS Document, from a Digital Signature XML Signature part to the Digital Signature Certificate part.	9.1.1	×	
O2.12	Digital Signature Definitions parts MAY be included in an XPS Document, from a FixedDocument part to the Digital Signature Definitions part.	9.1.1	×	
O2.13	DiscardControl relationships MAY be included in an XPS Document, from the package to a DiscardControl part.	9.1.1	×	
O2.14	DocumentStructure relationships MAY be included in an XPS Document, from a FixedDocument part to the DocumentStructure part.	9.1.1	×	
O2.15	PrintTicket relationships MAY be included in an XPS Document, from a FixedDocumentSequence, FixedDocument, or FixedPage part to a PrintTicket part.	9.1.1	×	
O2.16	StoryFragments relationships MAY be included in an XPS Document, from a FixedPage part to a StoryFragments part.	9.1.1	×	

O2.17	Thumbnail relationships MAY be included in an XPS Document, from the package to an Image part or from a FixedPage part to an Image part.	9.1.1	×	
O2.18	Color Profiles MAY be embedded in image files.	9.1.5	×	
O2.19	Thumbnail images MAY be attached to a FixedPage part using a Thumbnail relationship.	9.1.6	×	
O2.20	Fonts MAY be subsetted based on glyph usage.	9.1.7.1	×	
O2.21	Producers MAY use a 128-bit random number instead of a true GUID for an obfuscated font name.	9.1.7.3	×	
O2.22	An obfuscated Font part MAY have an arbitrary extension.	9.1.7.3	×	
O2.23	Producers MAY add digital signature requests and instructions to an XPS Document in the form of signature definitions.	9.1.10	×	
O2.24	A producer MAY sign against an existing signature definition to provide additional signature information.	9.1.10	×	
O2.25	A recipient of an XPS Document MAY also sign it against a signature definition.	9.1.10	×	
O2.26	This requirement was removed prior to Edition 1 of this specification.			
O2.27	Consumers MAY provide an algorithmic construction of the structure of an XPS Document based on a page-layout analysis, provided such structure is not explicitly provided in DocumentStructure and StoryFragments parts.	9.1.11		×
O2.28	A resource that is intended to be used across multiple fixed documents MAY be named according to the guidelines for shared resources.	9.2	×	
O2.29	Producers MAY include Markup Compatibility and Extensibility elements and attributes in DocumentStructure, FixedDocument, FixedDocumentSequence, FixedPage, Relationships, Remote Resource Dictionary, SignatureDefinitions, and StoryFragments parts.	9.3.1	×	
O2.30	Wherever a single whitespace character is allowed in XPS Document markup, multiple whitespace characters MAY be used (unless explicitly restricted by a pattern restriction in the corresponding schema).	9.3.4	×	
O2.31	Attributes in XPS Document markup that specify comma-delimited attribute values MAY, unless specified otherwise, OPTIONALLY include whitespace characters preceding or following the comma.	9.3.4	×	
O2.32	Where the XPS Document schema specifies attributes of types that allow whitespace collapsing, leading and trailing whitespace in the attribute value MAY be used along with other whitespace that relies on the whitespace collapsing behavior specified in the XML Schema Specification.	9.3.4	×	

I.5 Text

I.5.1 MUST Conformance Requirements

Table I–10. Text MUST conformance requirements

ID	Rule	Reference	Producer	Consumer
M5.1	If the CaretStops attribute is missing from the <Glyphs> element, a consumer MUST interpret the text as having a caret stop between each Unicode UTF-16 code unit and at the beginning and end of the text.	12.1		×[S]
M5.2	If the UnicodeString attribute of the <Glyphs> element specifies an empty string ("" or "{}") and the Indices attribute is not specified or is empty, the consumer MUST generate an error.	12.1	×	v
M5.3	The x:Key attribute of the <Glyphs> element MUST be present when the element is defined in a resource dictionary. It MUST NOT be specified outside a resource dictionary.	12.1	×	v
M5.4	The sum of the code unit counts for all the GlyphMapping entries in the Indices attribute MUST NOT exceed the number of UTF-16 code units in the UnicodeString attribute if the UnicodeString attribute is specified and does not contain an empty value ("" or "{}"). If a ClusterMapping is not specified within a GlyphMapping entry, the code unit count is 1. ~~If the Indices attribute specifies a GlyphIndex that does not exist in the font, the consumer MUST generate an error.~~	12.1.3	×	v
M5.5	If there is not a one-to-one mapping between code units in the UnicodeString attribute and the glyph indices, the GlyphIndex value in the Indices attribute MUST be specified.	12.1.3	×	
M5.6	The AdvanceWidth of the Indices attribute MUST be calculated as the exact unrounded origin of the subsequent glyph minus the sum of the calculated (that is, rounded) advance widths of the preceding glyphs.	12.1.3	×	
M5.7	A UnicodeString attribute value that begins with an open brace ("{") MUST be escaped with a prefix of "{}". If a UnicodeString attribute value starts with "{}", consumers MUST ignore those first two characters in processing the UnicodeString and in calculating index positions for the characters of the UnicodeString.	12.1.4	×	×
M5.8	This requirement was removed prior to Edition 1 of this specification.			
M5.9	If the UnicodeString attribute contains a Unicode code unit that cannot be mapped to a glyph index via a cmap table in the font and there is no corresponding GlyphIndex entry in the Indices attribute, the consumer MUST display the .notdef glyph	12.1.4	~~×~~	×
M5.10	In the absence of entries in the Indices attribute to override the Unicode code units in the UnicodeString attribute value, consumers MUST treat Unicode control marks in the UnicodeString attribute like ordinary characters and render the glyphs to which the Unicode	12.1.4		×

	control marks are mapped in the CMAP table.			
M5.11	Because advance-widths, glyph indices, and caret-stops are associated with the generated Unicode string, consumers MUST NOT normalize the UnicodeString attribute value to produce an internal representation.	12.1.4		×
M5.12	Producers MUST lay out algorithmically emboldened glyphs using advance widths that are 2% of the em size larger than when not algorithmically emboldened.	12.1.5	×	
M5.13	Consumers MUST implement the effect of algorithmic emboldening such that the black box of the glyph grows by 2% of the em size. When advance widths are omitted from the markup and the glyphs are algorithmically emboldened, the advance widths obtained from the horizontal metrics font table (if IsSideways is false) or the vertical metrics font table (if IsSideways is true) of the font MUST be increased by 2% of the em size.	12.1.5		×
M5.14	Producers MUST lay out algorithmically italicized glyphs using exactly the same advance widths as when not algorithmically italicized.	12.1.5	×	
M5.15	Producers MUST NOT specify text that is both right-to-left (BidiLevel attribute value of 1) and vertical (IsSideways attribute set to true).	12.1.6.2	×	v
M5.16	If a consumer does not understand the specified device font name, it MUST render the embedded version of the font.	12.1.7		×[P]
M5.17	When rendering a printer device font, consumers MUST use the UnicodeString attribute and ignore the glyph index components of the Indices attribute.	12.1.7		×[FP]
M5.18	When rendering a printer device font, consumers MUST still honor the advance width and x,y offset values present in the Indices attribute.	12.1.7		×[P]
M5.19	For producers, a <Glyphs> element with a specified device font name MUST have exactly one Indices glyph per character in the UnicodeString attribute. Its Indices attribute MUST NOT include any cluster specifications. If the Indices attribute includes a cluster mapping, the consumer MUST NOT use the device font name and MUST render the embedded version of the font.	12.1.7	×	v[FP]
M5.20	For producers of a <Glyphs> element with a specified device font name, each of the Indices glyphs MUST include a specified advance width and MUST include specified x and y offset values if they are non-zero.	12.1.7	×	
M5.21	This requirement was removed prior to Edition 1 of this specification.			
M5.22	If there are insufficient flags in the CaretStops attribute value to correspond to all the UTF-16 code units in the UnicodeString attribute value, all remaining UTF-16 code units in the Unicode string MUST be considered valid caret stops.	12.1.9		×[S]
M5.23	If the Indices attribute is specified, the values provided MUST be used in preference to values determined from the UnicodeString attribute alone.	12.1.3		×
M5.24	If the Indices attribute specifies a GlyphIndex that does not exist in	12.1.3	×	v

	the font, the consumer MUST generate an error.

I.5.2 SHOULD Conformance Requirements

Table I–11. Text SHOULD conformance requirements

ID	Rule	Reference	Producer	Consumer
S5.1	The value of the CaretStops attribute SHOULD indicate that the caret cannot stop in front of most combining marks and the second UTF-16 code unit of UTF-16 surrogate pairs.	12.1	×	
S5.2	If producers include control marks in the Unicode string, they SHOULD include an Indices attribute to specify glyph indices and/or character-to-glyph mapping information for the control marks.	12.1.4	×	
S5.3	If alternate vertical character representations are available in the font, the producer SHOULD use those in preference to the IsSideways attribute and provide their glyph indices in the Indices attribute.	12.1.6	×	
S5.4	Producers SHOULD NOT produce markup that will result in different rendering between consumers using the embedded font to render and consumers using the device font to render.	12.1.7	×	
S5.5	Specifying a UnicodeString for <Glyphs> elements is RECOMMENDED, as it supports searching, selection, and accessibility.	12.1.4	×	

I.5.3 OPTIONAL Conformance Requirements

Table I–12. Text OPTIONAL conformance requirements

ID	Rule	Reference	Producer	Consumer
O5.1	Producers MAY include Unicode control marks in the Unicode string. Such marks include control codes, layout controls, invisible operators, deprecated format characters, variation selectors, non-characters, and specials, according to their definition within the Unicode specification.	12.1.4	×	
O5.2	Producers MAY choose to generate UnicodeString attribute values that are not normalized by any Unicode-defined algorithm.	12.1.4	×	
O5.3	Consumers that understand the device font name MAY ignore the embedded font and use the device-resident version.	12.1.7		×[P]
O5.4	Glyph indices MAY be omitted from markup where there is a one-to-one mapping between the positions of Unicode scalar values in the UnicodeString attribute and the positions of glyphs in the glyph string and the glyph index is the value in selected character mapping table of the font.	12.1.10.1	×	
O5.5	Glyph advance widths MAY be omitted from markup where the advance width desired is specified in the font tables, once adjusted for	12.1.10.2	×	

M7.3	An xml:lang attribute within a resource definition MUST be interpreted in the context of the resource reference, not the resource definition.	14.2.3	×	×
M7.4	A remote resource dictionary MUST follow the requirements that apply to inline resource dictionaries.	14.2.3.1	×	v
M7.5	A remote resource dictionary MUST NOT contain any resource definition children that reference another remote resource dictionary.	14.2.3.1	×	v
M7.6	A <ResourceDictionary> element that specifies a remote resource dictionary in its Source attribute MUST NOT contain any resource definition children.	14.2.3.1	×	v
M7.7	Inline references to fonts or images in remote resource dictionary entries MUST be interpreted with the same base URI as the Remote Resource Dictionary part, not from the base URI of the part referring to the particular remote resource dictionary entry.	14.2.3.1	×	×
M7.8	When a resource definition references a previously defined resource with the same name in an ancestor resource dictionary, the reference MUST be resolved before the redefined resource is added to the dictionary	14.2.5	×	×
M7.9	If a resource definition references another resource, the reference MUST be resolved in the context of the resource definition, not in the context of the resource use.	14.2.5	×	×
M7.10	If a resource dictionary contains Markup Compatibility and Extensibility elements and attributes, the processing of the Markup Compatibility and Extensibility markup MUST occur in the context of the definition of the resource dictionary, not in the context of resource references.	14.2.6	×	×
M7.11	The x:Key attribute of the <MatrixTransform> element MUST be present when the element is defined in a resource dictionary. It MUST NOT be specified outside a resource dictionary.	14.4.1	×	v

I.7.2 OPTIONAL Conformance Requirements

Table I–15. Common properties OPTIONAL conformance requirements

ID	Rule	Reference	Producer	Consumer
O7.1	Resource dictionaries MAY be specified in separate parts (called remote resource dictionaries) and referenced from within the <FixedPage.Resources> or <Canvas.Resources> property element.	14.2	×	
O7.2	A resource definition MAY reference another resource defined prior to the point of reference, including a resource previously within the same resource dictionary.	14.2.3	×	
O7.3	If the resource dictionary does not appear in a separate part, a resource definition MAY reference a previously defined resource in a resource dictionary of a parent or ancestor <Canvas> or <FixedPage> element.	14.2.3	×	
O7.4	This requirement was removed prior to Edition 1 of this specification.			

I.12 Additional Conformance Requirements

I.12.1 MUST Conformance Requirements

Table I–28. Additional MUST conformance requirements

ID	Rule	Reference	Producer	Consumer
M12.1	FixedDocument parts MUST be referenced by <DocumentReference> elements within the FixedDocumentSequence part in ascending order. If additional FixedDocument parts are inserted into a fixed document sequence, producers MUST NOT unintentionally change the order of the existing FixedDocument part references.	–	×	
M12.2	A FixedDocument part MUST NOT be referenced more than once by a FixedDocumentSequence part.	–	×	v
M12.3	A FixedPage part MUST NOT be referenced more than once *in total*, throughout all FixedDocument parts.	–	×	v
M12.4	FixedPage parts MUST be referenced by <PageContent> elements within a fixed document in ascending order. If additional FixedPage parts are inserted into a FixedDocument part, producers MUST NOT unintentionally change the order of the existing FixedPage part references. Documents in languages for which the reading order of pages is back-to-front can be accommodated by adding <PageContent> elements to the FixedDocument in reverse order or by binding the right side of the page.	–	×	
M12.5	Any FixedDocumentSequence, FixedDocument, or FixedPage part that is reachable from the primary fixed payload root or its related parts by relationship or by the Source attribute on a <DocumentReference> or <PageContent> element MUST have no more than one attached PrintTicket part.	–	×	v
M12.6	Every Font part reachable from the primary fixed payload root or its related parts by relationship or by the Source attribute on a <DocumentReference> or <PageContent> element MUST be a valid OpenType font.	–	×	v
M12.7	The content types defined in this specification MUST NOT include parameters. A consumer MUST treat the presence of parameters on these content types as an error when the affected part is accessed.	**I.2**	×	v

End of informative text.

J. Bibliography

Independent JPEG Group. http://www.ijg.org/files/

A Nonaliasing, Real-Time Spatial Transform Technique. Fant, Karl M. *IEEE Computer Graphics and Applications* 6 (Jan. 1986): 71–80.

OS/2 and Windows Metrics. Microsoft Corporation. 2001. http://www.microsoft.com/typography/otspec/os2.htm

OpenType Font File. Microsoft Corporation. 2001. http://www.microsoft.com/typography/otspec/otff.htm

OpenType Specification, Version 1.4. Microsoft Corporation. 2004. http://www.microsoft.com/typography/otspec/default.htm

Print Schema. Microsoft Corporation. 2006. http://windowssdk.msdn.microsoft.com/en-us/library/default.aspx

~~*TIFF, Revision 6.0.* Adobe Systems Incorporated. 1992. http://partners.adobe.com/public/developer/en/tiff/TIFF6.pdf~~

Windows Color System in Windows Longhorn, WinHEC 2005 Version. Microsoft Corporation. 2005. http://download.microsoft.com/download/5/D/6/5D6EAF2B-7DDF-476B-93DC-7CF0072878E6/WCS.doc

Windows Media Photo Microsoft Corporation. http://www.microsoft.com/xps