XML Paper Specification

XPS Specification and Reference Guide

Base Document, Working Draft 1.0.1
Submitted to Ecma TC46 by Microsoft
September 2007

This document is "a work in progress". It is the starting point of discussions within Ecma Technical Committee TC46. It was adapted from the original XPS specification as published by Microsoft, to match standards editing style guidelines more closely, especially in the area of distinguishing between normative and informative text.

A number of formatting and textual issues remain to be addressed, as well as the bulk of the work of TC46 in amending the text to meet the scope and goals of the committee. Comments on this draft are very welcome; this early version is provided to enable the consistent use of clause and subclause references in those comments.

Contents

List of Figures

List of Tables

List of Examples

1. Scope

This specification defines *XPS*, the XML Paper Specification. XPS describes the set of conventions for the use of XML and other widely available technologies to describe the content and appearance of paginated documents. It is written for developers who are building systems that process XPS content.

A primary goal is to ensure the interoperability of independently created software and hardware systems that produce or consume XPS content. This specification defines the formal requirements that producers and consumers must satisfy in order to achieve interoperability.

This specification describes a paginated-document format called the *XPS Document*. The format requirements are an extension of the packaging requirements described in the Open Packaging Conventions (OPC) specification. That specification describes packaging and physical format conventions for the use of XML, Unicode, ZIP, and other technologies and specifications, to organize the content and resources that make up any document. They are an integral part of the XPS specification, and are included by reference.

Many XML-based building blocks within XPS make use of the conventions described in the Markup Compatibility specification that is relied upon by the OPC specification to facilitate future enhancement and extension of XPS markup. As such, that Markup Compatibility specification is included by reference.

2. Conformance

Language Notes

In this specification, the words that are used to define the significance of each requirement are written in uppercase. These words are used in accordance with their definitions in RFC 2119, and their respective meanings are reproduced below:

- MUST: This word, or the adjective "REQUIRED", means that the item is an absolute requirement of the specification.
- SHOULD: This word, or the adjective "RECOMMENDED", means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before choosing a different course.
- MAY: This word, or the adjective "OPTIONAL", means that this item is truly optional. [*Example*: One implementation may choose to include the item because a particular marketplace or scenario requires it or because it enhances the product. Another implementation may omit the same item. *end example*]

The words MUST NOT and SHOULD NOT are the negative forms of MUST and SHOULD, respectively. There is no negative form of MAY.

Software Conformance

Most requirements are expressed as format or package requirements rather than implementation requirements.

In order for consumers to be considered conformant, they must observe the following rules:

- They MUST NOT report errors when processing conforming instances of the documented formats except when forced to do so by resource exhaustion.
- They SHOULD report errors when processing non-conforming instances of the documented formats when doing so does not pose an undue processing or performance burden.

In order for producers to be considered conformant, they must observe the following rules:

- They MUST NOT generate any new, non-conforming instances of a documented format.
- They MUST NOT introduce any non-conformance when modifying an instance of a documented format.

Editing applications are subject to all of the above rules.

Comment [rcj1]: rules for consumers and producers.

Conformance requirements are documented inline in this specification, and each requirement is denoted by a letter (M – MUST; S – SHOULD; O – OPTIONAL) and a rule number, all enclosed in brackets ([…]). [*Example*: [M1.2] is a MUST requirement, [S2.4] is a SHOULD requirement, and [O3.9] is a MAY requirement.

end example] These rules are collected in I, “Conformance Requirements” for convenient reference.

3. Normative References

The following normative documents contain provisions, which, through reference in this text, constitute provisions of this specification. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this specification are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ECMA-376, "Office Open XML File Formats" (December 2006), Part 2, "Open Packaging Conventions", which is commonly referred to as OPC.

ECMA-376, "Office Open XML File Formats" (December 2006), Part 5, "Markup Compatibility and Extensibility".

ISO/IEC 2382.1:1993, Information technology — Vocabulary — Part 1: Fundamental terms.

ISO/IEC 10646:2003 (all parts), Information technology — Universal Multiple-Octet Coded Character Set (UCS).

4. Definitions

For the purposes of this specification, the following definitions apply. Terms explicitly defined in this specification are not to be presumed to refer implicitly to similar terms defined elsewhere.

alpha blending — Transparently blending two elements when rendering.

broken digital signature — A digital signature that conforms to the XPS Document signing rules but does not meet the digital signature validity requirements due to incorrect hash calculation or similar problems.

Compliant digital signature — A digital signature that conforms to the signing rules described in the XPS Document signing policy, regardless of signature validity.

Comment [rcj2]: Never used

consumer — A piece of software or a device that reads XPS packages.

content structure — The set of markup elements that allow expression of well-understood semantic blocks such as paragraphs, tables, lists, and figures.

content type — Describes the type of content stored in a part. Content types define a media type, a subtype, and an optional set of parameters, as defined in RFC 2045.

contour intersection point — The intersection of the flat line ending a dash and the contour of the shape.

device — A piece of hardware, such as a printer or scanner, that performs a single function or a set of functions.

document content — A document structural concept that identifies each block of individually readable content in an XPS Document.

document outline — A document structural concept that contains a structured index of the content in an XPS Document, much like a table of contents.

driver — A producer that has specific knowledge of the consumer of the XPS Document.

effective coordinate space — The default coordinate space (X,Y in the upper-left corner, units of 1/96") as modified by any RenderTransform or Transform attributes of the current element and any ancestor elements.

fixed payload — A payload that is rooted with a FixedDocumentSequence part.

fixed payload root — The root of a fixed payload is the FixedDocumentSequence part.

FixedDocument part — A common, easily indexed root for all pages within the document.

FixedDocumentSequence part — The part that assembles a set of FixedDocument parts within the fixed payload.

FixedPage part — The part that contains all of the visual elements to be rendered on a page.

Incompliant digital signature — A digital signature that does not conform to the XPS Document signing rules.

interleaved ordering — The layout style of a physical package where parts are broken into pieces and "mixed-in" with pieces from other parts. When delivered, interleaved packages help improve the performance of the consumer processing the package.

named color — An industry-defined color specification that identifies a particular color in a well-defined color schema, usually for purposes of printing.

named element — An element in the document structure markup that refers to an element in the fixed-page markup with a specified name.

package — A logical entity that holds a collection of parts.

package model — Defines a package abstraction that holds a collection of parts.

package relationship — A relationship whose target is a part and whose source is the package as a whole. Package relationships are found in the package relationships part named "/_rels/.rels".

part — A stream of bytes with a MIME content type and associated common properties. Typically corresponds to a file (as on a file system), a stream (as in a compound file), or a resource (as in an HTTP URI).

part name — A part name is used to refer to a part in the context of a package, typically as part of a URI. The part name is, by definition, the path component of a pack URI.

payload — A complete collection of interdependent parts and relationships within a package.

physical imageable size — Represents the area of a page that is printable by a specific device.

physical media size — Represents the physical media on which the content will be printed.

physical model — Defines the mapping between the components of the package model to the features of a particular physical format based on the ZIP specification.

piece — A portion of a part. Pieces of different parts may be interleaved together. The individual pieces are named using a unique mapping from the part name. The piece name grammar does not conform to the part name grammar. Pieces are not addressable in the package model.

primary fixed payload root — The fixed payload root that is referenced by the XPS package StartPart relationship.

PrintTicket part — A PrintTicket part provides the settings used when a package is printed. PrintTicket parts may be attached to the entire package, or at lower levels in the structure, such as individual pages.

producer — A piece of software or a device that writes XPS packages.

producer bleed size — Represents the overflow (or "bleed") box used by the producer for registration and layout.

producer content size — Represents the content bounding box specified by the producer.

producer media size — Represents the physical media on which the content will be printed.

property — A characteristic of a markup element, referred to as an attribute of the element.

property attribute — XPS Document property values can be expressed as either property attributes or property elements.

property element — An XPS Document property value can be expressed as either a property attribute or a property element.

property value — The value of a property, expressed as an XML attribute, an XML child element, or an entry in the resource dictionary.

questionable digital signature — A digital signature that conforms to the XPS Document signing rules but has a problem during validation of the signature such as the inability to contact the certificate authority to validate its authenticity or the markup contains markup compatibility elements and attributes that may change the representation of the signed content.

relationships — A relationship represents the kind of connection between a source part and a target part in a package. Relationships make the connections between parts directly discoverable without looking at the content in the parts, and without altering the parts themselves. See also, package relationship.

relationships part — A part containing an XML representation of relationships.

remote resource dictionary — A part containing a resource dictionary.

required part — A part, such as an image or font, that is referenced from other parts, and is required for valid processing of the referencing part.

resource definition — A shareable property value, with a name, defined within a resource dictionary. Any property value defined by fixed page markup may be held in a resource dictionary. Each resource definition has a key that is unique within the scope of the resource dictionary.

resource dictionary — A resource dictionary holds resources. Each resource in a resource dictionary carries a name. The resource's name can be used to reference the resource from a property's XML attribute.

resource reference — An attribute whose value refers to an entry in a resource dictionary. Resource references appear in the format "{StaticResource RscName}" where RscName corresponds to a matching entry in the resource dictionary with an x:Key attribute value.

signature definition — The means by which XPS Document authors provide co-signature requirements and workflow-specific signature information.

signature spot — A visual element that indicates that a digital signature has been applied or requested.

signing rules — The set of rules that define whether a particular digital signature is compliant with the XPS Document **signature** policy.

simple ordering — Simple ordering the parts in the package are laid out with a defined ordering. When such a package is delivered in a purely linear fashion, starting with the first byte in the package through to the last that, all of the bytes for the first part arrive first, then all of the bytes for the second part, and so on.

stories — Meaningful semantic blocks that provide structure to markup elements that appear in a FixedPage part.

story — A block of individually readable content in an XPS Document.

story fragment — A portion of a story that appears within the scope of a single fixed page.

stream — A linearly ordered sequence of bytes.

thumbnail — Images that help end-users identify parts of a package or a package as a whole.

valid digital signature — A digital signature that conforms to the XPS Document signing rules and is not a broken digital signature or questionable digital signature.

XPS Document — A package that contains a discoverable fixed payload and is a format for storing paginated documents defined by the XPS specification.

XPS Document StartPart relationship — The specific relationship type that identifies the root of a fixed payload within an XPS Document.

ZIP Archive — A physical ZIP file that is displayed by the file system. A ZIP archive contains **ZIP items**.

5. Notational Conventions

Document Conventions

Except where otherwise noted, syntax descriptions are expressed in the ABNF format as defined in RFC 4234.

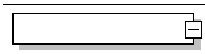
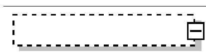
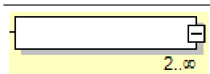
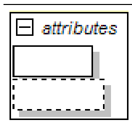
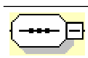
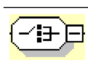
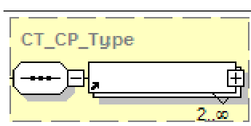
Definition terms are formatted like *this*.

Syntax descriptions and code are formatted in `monospace` type.

Replaceable items are formatted in *`monospace cursive`* type.

Diagram Notes

In some cases, markup semantics are described using diagrams. The diagrams place the parent element on the left, with attributes and child elements to the right. The symbols are described below.

Symbol	Description
	Required element. This box represents an element that MUST appear exactly once in markup when the parent element is included. The "+" and "–" symbols on the right of these boxes have no semantic meaning.
	Optional element. This box represents an element that can appear zero or one times in markup when the parent element is included.
	Range indicator. These numbers indicate that the designated element or choice of elements can appear in markup any number of times within the range specified.
	Attribute group. This box indicates that the enclosed boxes are each attributes of the parent element. Solid-border boxes are required attributes; dashed-border boxes are optional attributes.
	Sequence symbol. The element boxes connected to this symbol can appear in markup in the illustrated sequence only, from top to bottom.
	Choice symbol. Only one of the element boxes connected to this symbol can appear in markup.
	Type indicator. The elements within the dashed box are of the complex type indicated.

6. Acronyms and Abbreviations

This clause is informative

The following acronyms and abbreviations are used throughout this specification:

IEC — the International Electrotechnical Commission

ISO — the International Organization for Standardization

W3C — World Wide Web Consortium

End of informative text

7. General Description

This specification is intended for use by implementers, academics, and application programmers. As such, it contains explanatory material that, strictly speaking, is not necessary in a formal specification.

This specification is divided into the following subdivisions:

1. Front matter (clauses 1–7).
2. XPS Documents (clauses 8–18), which presents the details of the primarily XML-based XPS Document format. These clauses describe the XML markup that defines the composition of documents and the appearance of each page. They also include rendering rules that enable devices and applications to display and print XPS Documents with full fidelity in a wide range of environments and scenarios.
3. XPS Document Markup Reference (clause 19), which presents a consolidated reference of XPS Document markup elements and their attributes.
4. Annexes (clauses A–J), which contain additional technical details and schemas, as well as convenient reference information.

Examples are provided to illustrate possible forms of the constructions described. References are used to refer to related clauses. Notes are provided to give advice or guidance to implementers or programmers. Annexes provide additional information or summarize the information contained elsewhere in this specification.

Clauses 1–5 and 7–19, and annexes A–H and J, form a normative part of this specification; and the clause 6, annexe I, examples, notes, and the index, are informative.

Except for whole clauses or annexes that are identified as being informative, informative text that is contained within normative text is indicated in the following ways:

1. Examples within narrative are indicated as follows: [*Example*: … *end example*]
2. Examples of XML are indicated as follows: *Example m.n*: caption … *end example*]
3. [*Note*: … *end note*]

8. XPS Document Format

This specification describes how the XPS Document format is organized internally and rendered externally. It is built upon the principles described in the OPC specification. XPS Documents MUST observe all conformance requirements and recommendations of that specification, except where indicated otherwise [M1.1]. The information presented here is intended both for producers, which emit content in the XPS Document format, and consumers, which access and render or process the contents of an XPS Document.

The XPS Document format represents a set of related pages with a fixed layout, which are organized as one or more *documents*, in the traditional meaning of the word. A file that implements this format includes everything necessary to render fully those documents on a display device or physical medium (such as paper). This includes all resources such as fonts and images that might be required to render individual page markings.

In addition, the format includes optional components that build on the minimal set of components required to render a set of pages. This includes the ability to specify print job control instructions, to organize the minimal page markings into larger semantic blocks such as paragraphs, and to rearrange physically the contents of the format for easy consumption in a streaming manner, among others.

Finally, the XPS Document format implements the common package features specified by the Open Packaging Conventions specification that support digital signatures and core properties.

8.1 How This Specification Is Organized

This subclause is informative

Clause	Description
Parts and Relationships (§9)	This clause describes how XPS Documents use the packaging model (as described in the OPC specification) to organize data. All part and relationship types are described in detail, including how they are used and what they can contain. This clause also describes the XPS Document markup model, in particular, its parts, and how the XML markup relates to the packaging conventions and recommendations it builds on.
Documents (§10)	The fundamental building blocks of the XPS Document format are described here. This clause describes how pages are composed into larger documents and how documents are composed into document sequences. These components are represented in markup.
Graphics (§11)	This is the first of several clauses that describe page markings, in particular, vector graphics. The concepts of paths, geometries, and figures are introduced. Vector graphics are represented in page-layout XML markup.
Text (§12)	This clause describes how to include text markings in page-layout markup. It describes how to reference a font and extract information from a font to render the page.
Brushes (§13)	Both vector graphics and text are rendered by applying any of the brushes described in this clause. This includes brushes that are created from solid colors, gradients, images, or other page-layout markup.

Clause	Description
Common Properties (§)14	Several page-layout markup elements share a common set of properties. These properties can be expressed either as XML attributes or as XML child or descendant elements. This clause describes these common properties.
Color (§15)	XPS Documents support a wide range of color options and color spaces, both for vector and raster images. This clause describes the combinations of image formats and color markup that may be used. A number of color-related topics are discussed, including color separation, color profiles, and color blending.
Document Structure and Interactivity (§16)	This clause describes the components of the XPS Document format that support assigning larger semantic meaning to individual page markings in order. [*Example*: Such markings might be tables or paragraphs. *end example*] It also provides a mechanism to describe an outline of the document. Additionally, this clause provides guidance on how consumers that enable interactive features such as hyperlinks, selection, and accessibility tools should use the format. It also describes how producers should emit content to enable interactive features.
XPS Document Package Features (§17)	This clause describes how package features (as described in the OPC specification) are used and extended in the XPS Document format. This includes interleaving, digital signatures, and core properties.
Rendering Rules (§18)	This clause provides precise instructions for rendering XPS Document contents to ensure a consistent result among various implementations.
Elements (§19)	The full list of elements described throughout the preceding clauses is assembled in this clause, in alphabetical order, for easy reference.
Signature Definitions Schema (§A)	This annex includes the XSD schema for the Signature Definitions part.
XPS Document Schema (§B)	This annex includes the XSD schema for the FixedDocumentSequence, FixedDocument, and FixedPage parts.
Resource Dictionary Key Schema (§C)	This annex provides the XSD schema for the resource dictionary **Key** attribute, used by several elements in the XPS Document schema.
Document Structure Schema (§D)	This annex provides the XSD schema for the DocumentStructure and StoryFragments parts.
Discard Control Schema (§E)	This annex includes the XSD schema for the DiscardControl part for interleaving.
Abbreviated Geometry Syntax Algorithm (§F)	This annex provides a sample algorithm for interpreting the abbreviated geometry syntax provided to succinctly describe geometric regions in a single attribute.

Clause	Description
scRGB Gamut Boundary Definition (§G)	The scRGB color space's gamut boundary used by this specification is defined in this annex.
Standard Namespaces and Content Types (§H)	This annex defines all of the XML namespace names, content types, and relationship types used by all XPS Document parts and relationships.
Conformance Requirements (§I)	This annex assembles all the conformance requirements specified throughout the previous clauses and annexes into a comprehensive list for reference purposes.

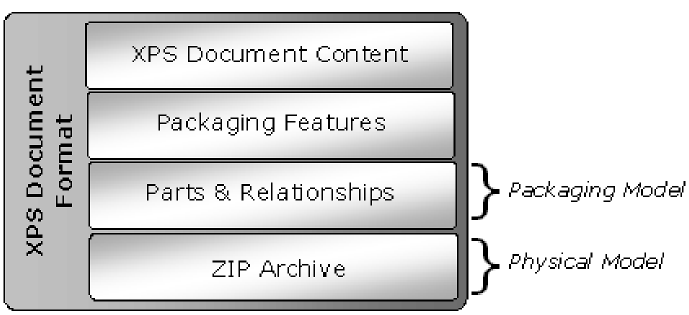
End of informative text

8.2 Package

The XPS Document format MUST use a ZIP archive for its *physical model* [M1.2]. The OPC specification describes a packaging model; that is, how the package is represented internally with parts and relationships.

The XPS Document format includes a well-defined set of parts and relationships, each fulfilling a particular purpose in the document. The format also extends the package features, including digital signatures, thumbnails, and interleaving.

Figure 8–1. Package-based XPS Document format



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. There can be more than one fixed payload in an XPS Document.

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].

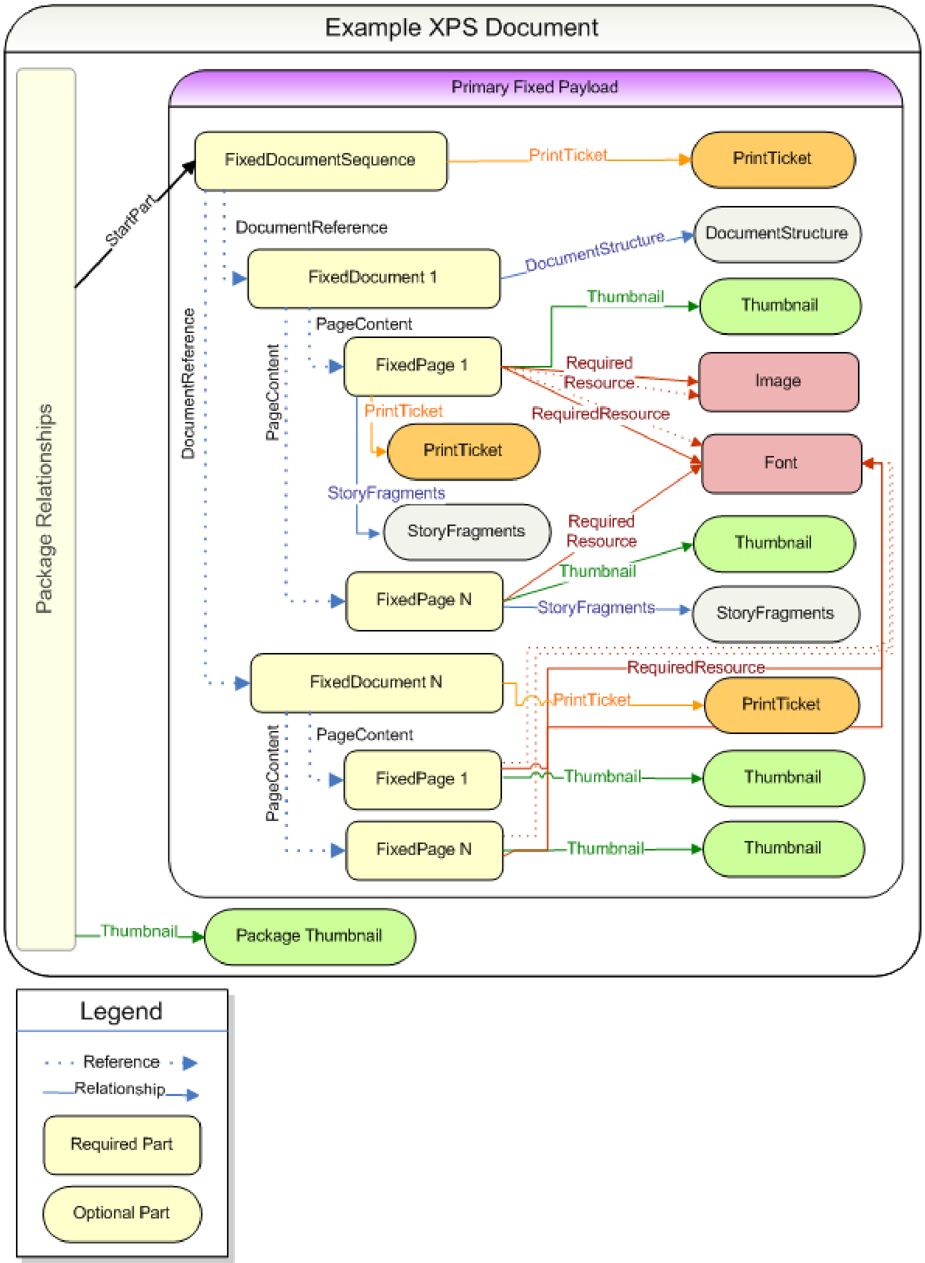
The payload includes the full set of parts required for processing the FixedDocumentSequence part. All content to be rendered MUST be contained in the XPS Document [M2.1]. The parts that can be found in an XPS Document are listed in Table 9–1. Relationships and content types for these parts are defined in H, "Standard Namespaces and Content Types." 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]
Image (§9.1.5) JPEG image (§9.1.5.1) PNG image (§9.1.5.2) TIFF image (§9.1.5.3) Windows Media Photo image (§9.1.5.4)	References an image file.	REQUIRED if an <ImageBrush> element is present [M2.7]
Remote resource dictionary (§9.1.8)	Contains a resource dictionary for use by fixed page markup.	REQUIRED if a key it defines is referenced [M2.8]
Thumbnail (§)9.1.6	Contains a small JPEG or PNG image that represents the contents of the page or package.	OPTIONAL [O2.1]
PrintTicket (§9.1.9)	Provides settings to be used when printing the package.	OPTIONAL [O2.2]
ICC profile (§??)	Contains an ICC Version 2 color profile optionally containing an embedded Windows Color System (WCS) color profile.	OPTIONAL [O2.3]
DocumentStructure (§9.1.11)	Contains the document outline and document contents (story definitions) for the XPS Document.	OPTIONAL [O2.4]
StoryFragments (§)9.1.12	Contains document content structure for a fixed page.	OPTIONAL [O2.5]
SignatureDefinitions (§9.1.10)	Contains a list of digital signature spots and signature requirements.	OPTIONAL [O2.6]
DiscardControl (§??)	Contains a list of resources that are safe for consumers to discard during processing.	OPTIONAL [O2.7]

Example 9–1. A typical XPS Document



end example]

9.1.1 Fixed Payload Relationships

Internal resources are associated with parts by relationships and inline references. XPS Documents MUST NOT reference external XPS resources [M2.1]. In general, inline resource references are represented inside the referring part in ways that are specific to the content type of the part, that is, in arbitrary markup or application-specific encoding. Relationships represent the type of connection between a source part and a target resource, and they allow parts to be related without modifying them. For more information, see the OPC specification.

Resources, which include fonts, images, color profiles, and remote resource dictionaries, that are referenced by inline URIs but are necessary to render the page MUST use the Required Resource relationship from the FixedPage part to the resource [M2.10]. If any resource references *other* resources, the producer MUST also use the Required Resource relationship from the FixedPage part to the indirectly referenced resource [M2.10].

It is RECOMMENDED that there be exactly *one* Required Resource relationship from the FixedPage part for each resource referenced from markup [S2.1]. Multiple Required Resource relationships from a FixedPage part to a resource are not considered an error, but they reduce efficiency. It is not considered an error if a FixedPage part that does not use a specific resource in its markup references the resource via a Required Resource relationship; however, doing so may reduce efficiency for consumers.

Relationship types are defined in H, "Standard Namespaces and Content Types."

Table 9–2. Fixed payload relationships

Name	**Description**	**Required/Optional**
Core Properties	Relationship from the package to the Core Properties part.	OPTIONAL [O2.8]
Digital Signature Origin	Relationship from the package to the Digital Signature Origin part.	OPTIONAL [O2.9]
Digital Signature	Relationship from the Digital Signature Origin part to an Digital Signature XML Signature part.	OPTIONAL [O2.10]
Digital Signature Certificate	Relationship from a Digital Signature XML Signature part to a Digital Signature Certificate part.	OPTIONAL [O2.11]
Digital Signature Definitions	Relationship from the FixedDocument part to a Digital Signature Definitions part.	OPTIONAL [O2.12]
DiscardControl	Relationship from the package to a DiscardControl part.	OPTIONAL [O2.13]
DocumentStructure	Relationship from the FixedDocument part to a DocumentStructure part.	OPTIONAL [O2.14]
PrintTicket	Relationship from a FixedDocumentSequence part, a FixedDocument part, or a FixedPage	OPTIONAL [O2.15]

	part to a PrintTicket part.	
Required Resource	Relationship from a FixedPage part to a required resource, including Font, Image, ColorProfile, and Remote Resource Dictionary parts. Required resources can be shared between pages.	REQUIRED for each resource referenced from a FixedPage [M2.10]
Restricted Font	Relationship from a FixedDocument part to a Font part. Specifies the referenced font as restricted, disallowing any modification or editing of any <Glyphs> element text using the referenced font.	REQUIRED for each preview and print font used [M2.12]
StartPart	Relationship from the package to the FixedDocumentSequence part that is the primary fixed payload root.	REQUIRED [M2.13, M2.14]
StoryFragments	Relationship from a FixedPage part to the StoryFragments part for the page.	OPTIONAL [O2.16]
Thumbnail	Relationship from the package to an Image part or from a FixedPage part to an Image part.	OPTIONAL [O2.17]

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 [M2.77].

9.1.2 FixedDocumentSequence Part

The *FixedDocumentSequence part* assembles a set of fixed documents within the fixed payload. [*Example*: A printing client can assemble two separate documents, a two-page cover memo and a twenty-page report (both are FixedDocument parts), into a single package to send to the printer. *end example*]

The FixedDocumentSequence part is the only valid root of a fixed payload. Even if an XPS Document contains only a single fixed document, the FixedDocumentSequence part is still used. One FixedDocumentSequence part per fixed payload is REQUIRED [M2.3].

Fixed document sequence markup specifies each fixed document in the fixed payload in sequence, using <DocumentReference> elements. The order of <DocumentReference> elements determines document order and MUST be preserved by editing consumers [M2.15]. Each <DocumentReference> element SHOULD reference a FixedDocument part by relative URI [S2.2]. For more information, see §10.1, "<FixedDocumentSequence> Element," on page 49.

The content type of the FixedDocumentSequence part is defined in H, "Standard Namespaces and Content Types."

9.1.3 FixedDocument Part

The *FixedDocument part* is a common, easily indexed root for all pages within the document. A fixed document identifies the set of fixed pages for the document.

The markup in the FixedDocument part specifies the pages of a document in sequence using <PageContent> elements. The order of <PageContent> elements determines page order and MUST be preserved by editing consumers [M2.16]. Each <PageContent> element SHOULD reference a FixedPage part by relative URI [S2.3]. For more information, see §10.2, "<FixedDocument> Element," on page 50.

The content type of the FixedDocument part is defined in H, "Standard Namespaces and Content Types."

9.1.4 FixedPage Part

The *FixedPage part* contains all of the visual elements to be rendered on a page. Each page has a fixed size and orientation. The layout of the visual elements on a page is determined by the fixed page markup. This applies to both graphics and text, which are represented with precise typographic placement. The contents of a page are described using a powerful but simple set of visual primitives.

Each FixedPage part specifies the contents of a page within a <FixedPage> element using <Path> and <Glyphs> elements (using various brush elements) and the <Canvas> grouping element. The <ImageBrush> and <Glyphs> elements (or their child or descendant elements) can reference Image parts or Font parts by URI. They SHOULD reference these parts by relative URI [S2.4]. For more information, see §10.3, "<FixedPage> Element," on page 53.

The content type of the FixedPage part is defined in H, "Standard Namespaces and Content Types."

9.1.5 Image Parts

Image parts reference image files. A single image may be shared among multiple fixed pages in one or more fixed documents. Images referenced in markup MUST be internal to the package [M2.1]. References to images that are external to the package are invalid.

Images are included in XPS Documents with an <ImageBrush> element and an **ImageSource** attribute to reference a part with the appropriate content type. For more information, see §H.2, "Content Types," on page 406. Fixed pages MUST use a Required Resource relationship to each Image part referenced [M2.10]. For more information, see §H.3, "Relationship Types," on page 407.

XPS Documents support the following image formats:

- JPEG
- PNG
- TIFF
- Windows Media Photo

Color profiles MAY be embedded in image files [O2.18] and SHOULD be used by consumers [S2.5]. See 15, "Color," for more details.

9.1.5.1 JPEG Images

It is RECOMMENDED that JPEG image part names end with the extension ".jpg" [S2.6]. JPEG image parts MUST contain images that conform to the JPEG specification [M2.17]. Consumers SHOULD support JPEG images that contain APP0, APP2, APP13, and APP14 markers [S2.34]. Consumers MUST support JPEG images that contain the APP1 marker and interpret the EXIF color space correctly [M2.78].

Table 9–3. Supported JPEG APPn markers

APPn marker	Originating source
APP0	JFIF specification
APP1	EXIF extension defined by JEITA
APP2	ICC profile marker defined by the ICC specification
APP13	Photoshop 3.0 extension
APP14	Adobe DCT Filters in PostScript Level 2 extension

Some JPEG implementations have limited support for CMYK JPEG images, such as:

- CMYK is converted to RGB in the decoder using fixed tables instead of the supplied ICC profile.
- ICC Profiles embedded using APP2 are limited in length, because APPn marker chunking is not supported.

Therefore, the use of CMYK images is NOT RECOMMENDED in XPS Documents because rendering results can differ significantly between implementations. TIFF or Windows Media Photo images SHOULD be used instead to represent CMYK images [S2.7].

If both APP2 and APP13 markers are specified, the APP2 marker takes precedence. If the JPEG image is embedded in a TIFF image, the TIFF ICC profile settings are used.

9.1.5.2 PNG Images

It is RECOMMENDED that PNG image part names end with the extension ".png" [S2.8]. PNG image parts MUST contain images that conform to the PNG specification [M2.18].

Table 9–4. Support for ancillary PNG chunks

Chunk	Support Level
tRNS	MUST Support [M2.19]
iCCP	MUST Support [M2.20]
sRGB	MUST Ignore [M2.21]
cHRM	MUST Ignore [M2.22]
gAMA	MUST Ignore [M2.23]
sBIT	MUST Ignore [M2.24]

9.1.5.3 TIFF Images

It is RECOMMENDED that TIFF image part names end with the extension ".tif" [S2.9]. TIFF image parts MUST contain images that conform to the TIFF specification

[M2.25]. XPS Document consumers MUST support baseline TIFF 6.0 with some extensions, as noted in Table 9–5 [M2.26]. These tags MUST be supported for the specified image types [M2.26]. If consumers encounter a tag that is not included below, they SHOULD ignore that tag [S2.10].

Table 9–5. Supported TIFF 6.0 tags

Image type	**Tags**
Bilevel images	PhotometricInterpretation (0 and 1)
	Compression (1, 2, 3, 4, 5, or 32773)
	ImageLength
	ImageWidth
	ResolutionUnit (1, 2, or 3)
	RowsPerStrip
	StripByteCounts
	StripOffsets
	XResolution
	YResolution
Grayscale images	PhotometricInterpretation (0 and 1)
	BitsPerSample (4, 8, or 16)
	Compression (1, 5, 6, or 32773)
	ImageLength
	ImageWidth
	ResolutionUnit (1, 2, or 3)
	RowsPerStrip
	StripByteCounts
	StripOffsets
	XResolution
	YResolution
Palette color images	BitsPerSample (1, 4, or 8)
	ColorMap
	Compression (1, 5, or 32773)
	ImageLength
	ImageWidth
	PhotometricInterpretation (3)
	ResolutionUnit (1, 2, or 3)
	RowsPerStrip
	StripByteCounts
	StripOffsets
	XResolution
	YResolution
RGB images	BitsPerSample (8,8,8 or 16,16,16; *or* if ExtraSamples=1 or 2: 8,8,8,8 or 16,16,16,16)
	Compression (1, 5, 6, or 32773)

	ExtraSamples (0, 1, or 2) ICC Color Profile [tag 34675] ImageLength ImageWidth PhotometricInterpretation (2) PlanarConfiguration (1) ResolutionUnit (1, 2, or 3) RowsPerStrip SamplesPerPixel (3; *or* if ExtraSamples=1 or 2: 4) StripByteCounts StripOffsets XResolution YResolution
CMYK images (TIFF extension)	BitsPerSample (8,8,8,8 or 16,16,16,16; *or* if ExtraSamples=1 or 2: 8,8,8,8,8 or 16,16,16,16,16) Compression (1, 5, 6, or 32773) ExtraSamples (0, 1, or 2) ICC Color Profile [tag 34675] ImageLength ImageWidth InkSet (1) NumberOfInks (4) PhotometricInterpretation (5) PlanarConfiguration (1) ResolutionUnit (1, 2, or 3) RowsPerStrip SamplesPerPixel (4; *or* if ExtraSamples=1 or 2: 5) StripByteCounts StripOffsets XResolution YResolution

If the TIFF image contains multiple image file directories (IFDs), consumers MUST use only the first IFD and ignore all others [M2.27].

If the ResolutionUnit tag is set to 1 (no units), XResolution and YResolution are interpreted in the same manner as if the ResolutionUnit was set to 2 (inches).

If no color profile is embedded in the TIFF image or stored in a separate part associated with the TIFF image according to the mechanisms described in §15.2.8, "Images and Color Profile Association" on page 216, then the default color space MUST be treated as defined in §15.2.9, "Color Space Pixel Formats for Raster Images," on page 216 [M8.30].

The following features of the TIFF specification MUST be supported in addition to the tags described in Table 9–5:

- Baseline TIFF (Sections 1–10) with the exception of the following tags [M2.26]:
  - CellLength
  - CellWidth
  - GrayResponseCurve
  - GrayResponseUnit
  - MaxSampleValue
  - MinSampleValue
  - Orientation
  - Thresholding
- CCITT bilevel encodings (Section 11) [M2.28]
- CMYK images (Section 16) [M2.29]

- Associated alpha data (Section 18) [M2.30]
  - ExtraSamples tag value of 1: Treat alpha as pre-multiplied alpha (see §18.4.1, "Pre-Multiplied Alpha and Superluminous Colors," on page 281 for details)
  - ExtraSamples tag value of 2: Treat alpha as non-pre-multiplied alpha
- LZW compression (Section 13) [M2.31]
- Differencing predictors (Section 14) [M2.32]
- JPEG compression (Section 22)
  - Only compression mode 6 MUST be supported [M2.33]
- Embedded ICC Profile (described in the ICC specification) [M2.34]
- EXIF IFD (tag 34665) as described in the EXIF specification. The EXIF color space MUST be interpreted correctly [M2.79].

Consumers that support tags and features not described above can result in undesirable differences in the appearance of XPS Documents. Producers cannot rely on a consistent interpretation of tags or features that are not described above and therefore SHOULD NOT use any such tags or features [S2.10].

[*Note*: Many TIFF images in circulation today deviate from the TIFF Specification. Over time, TIFF-consuming implementations have developed a certain tolerance for such deviations by attempting to deduce the intent of the TIFF image author and correct for apparent errors or deviations. *end note*]

Therefore, XPS Document consumers SHOULD do the following [S2.11]:

- Test with as many different TIFF images as possible.
- Correct common mistakes in TIFF images, such as:
  - Not all BitsPerSample hold the same value
  - Number of BitsPerSample does not match SamplesPerPixel
  - PhotometricInterpretation 1 or 2 (instead of 3) used when BitsPerSample is set to "8,8,8"
  - Missing ExtraSamples tag, but SamplesPerPixel does not match the value expected by the PhotometricInterpretation tag, thus necessitating deduction of the ExtraSamples value.
- Implement a reasonable recovery strategy when a problematic TIFF image is encountered.

Comment [rcj3]: What is "reasonable"?

9.1.5.4 Windows Media Photo Images

It is RECOMMENDED that Windows Media Photo image part names end with the extension ".wdp" [S2.12]. Windows Media Photo image files MUST conform to the Windows Media Photo specification [M2.35]. XPS Documents support Windows Media Photo images with the characteristics identified in Table 9–6.

Table 9–6. Supported Windows Media Photo features

Color space	Pixel formats	Compression	Alpha
Grayscale	8-bit integer 16-bit integer 16-bit half-float* 16-bit fixed point* 32-bit fixed point*	Lossy Lossless	None
sRGB	8-bit integer 16-bit integer	Lossy Lossless	1-channel 1-channel pre-multiplied
scRGB	16-bit half-float 16-bit fixed point 32-bit IEEE float 32-bit fixed point RGBE-Radiance	Lossy Lossless	1-channel 1-channel pre-multiplied RGBE-Radiance (no alpha channel)
CMYK	8-bit integer 16-bit integer	Lossy Lossless	1-channel independent
N-channel (including named color N-tone)	8-bit integer 16-bit integer	Lossy Lossless	1-channel independent
Profiled RGB (3-channel)	8-bit integer 16-bit integer	Lossy Lossless	1-channel 1-channel pre-multiplied

* The value range of these formats is the same as scRGB.

If no color profile is embedded in the Windows Media Photo image or stored in a separate part associated with the Windows Media Photo image according to the mechanisms described in §15.2.8, "Images and Color Profile Association" on page 216, then the default color space MUST be treated as defined in §15.2.9, "Color Space Pixel Formats for Raster Images," on page 216 [M8.30].

9.1.6 Thumbnail Parts

Thumbnails are small images that represent the contents of a fixed page or an entire XPS Document. Thumbnails enable users of viewing applications to select a page easily.

Thumbnail images MAY be attached using a relationship to the FixedPage parts [O2.19]. Each FixedPage part MUST NOT have more than one thumbnail part attached [M2.36]. Relationships to thumbnail parts are defined in H, "Standard Namespaces and Content Types." It is RECOMMENDED that if thumbnails are used for pages, a thumbnail SHOULD be included for every page in the document [S2.13].

Comment [rcj4]: Does this mean a different thumbnail for each page? Clarify.

Although the OPC specification allows thumbnails to be attached to any part, XPS Document consumers SHOULD only process thumbnails associated via a package relationship from the package as a whole or via a relationship from a FixedPage part [S2.14]. These thumbnails MUST be in either JPEG or PNG format [M2.37]. Thumbnails attached to any other part SHOULD be ignored by XPS Document consumers [S2.14]. The content types of thumbnail parts are specified in §H.2, "Content Types," on page 406.

For more information about the relationship type for thumbnail parts, see §H.3, "Relationship Types," on page 407.

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, "Non-Standard Font Compatibility Encoding" for additional information) [S2.15].

Font parts are referenced using the **FontUri** attribute of the <Glyphs> element. A single font may 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]. [*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".

Content types for fonts differ depending on whether the font is non-obfuscated or obfuscated (see §9.1.7.2, "OpenType Font Embedding," on page 31). Content types are summarized in H, "Standard Namespaces and Content Types."

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, "Embedded Font Obfuscation," on page 33 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 may 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, "DeviceFontName Attribute," on page 115 and §H.3, "Relationship Types," on page 407 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.	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 are present in the font, MUST NOT embed the font [M2.51].	MUST NOT extract or install permanently [M2.52].
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:

4. Obfuscated font files are embedded within an XPS Document package in a form that cannot be directly installed on any client operating system.
5. 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.
6. 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. [*Note*: Microsoft implementations for XPS Documents always perform obfuscated font embedding and do not extract or permanently install the font [S2.19]. *end note*] However, there are vertical solutions in which implementations may 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, "Standard Namespaces and Content Types" [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:

   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].

Editing consumers MUST NOT edit a document where the FixedDocument part has a Restricted Font relationship [M2.43]. When invoking editing functionality, editing consumers 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].

Printing and display-only consumers MUST consider an XPS Document valid even if the producer failed to properly set the Restricted Font relationship [M2.12].

Comment [rcj5]: Are these subcategories of consumer that need defining?

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 are discouraged from using this non-standard encoding and they SHOULD write PUA codepoints to the **UnicodeString** attribute [S2.15].

Comment [rcj6]: Should this be wriiten using SHOULD NOT?

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, "Remote Resource Dictionaries," on page 177.

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 may 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.

9.1.9.2 Mapping PrintTicket Parts to Fixed Payload Parts

The PrintTicket defines a hierarchy of print settings to identify the applicability of a setting to different content levels within a print job. Specifically, the PrintTicket supports a hierarchy that is rooted in the print job. Multiple documents are derived from the print job, and multiple pages are defined from each document. A PrintTicket can be associated with each level in this hierarchy. These PrintTicket parts are labeled "job-level," "document-level," and "page-level," respectively.

Print settings are defined within the PrintTicket. Each print setting has a scoping prefix to indicate the level at which it applies. [*Example*: "JobInputBin" is a job-level setting, "DocumentStaple" is a document-level setting, and "PageOrientation" is a page-level setting. *end example*] Settings in level-specific PrintTicket parts are restricted by the scoping prefix. A level-specific PrintTicket MUST contain only settings scoped to the current level and child levels [M2.59]. 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 [M2.59].

Within an XPS Document, there is a direct mapping between the PrintTicket levels and the fixed payload parts:

- Consumers MUST process job-level, document-level and page-level settings of PrintTicket parts associated with FixedDocumentSequence parts [M2.60].
- Producers SHOULD only attach PrintTicket parts containing only document-level and page-level settings with FixedDocument parts [S2.22].
- 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 [M2.61].
- Producers SHOULD only attach PrintTicket parts containing only page-level settings with FixedPage parts [S2.23].
- 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 [M2.62].

PrintTicket parts are associated with parts via the PrintTicket relationship defined in H, "Standard Namespaces and Content Types."

9.1.9.3 Processing PrintTicket Parts

Printing consumers MUST process all PrintTicket parts within the XPS Document [M2.58].

When processing a PrintTicket, consumers MUST first remove all levels of PrintTicket content not applicable to the current element [M2.63] (see §9.1.9.2, "Mapping PrintTicket Parts to Fixed Payload Parts," on page 36).

Second, consumers MUST validate the PrintTicket according to the methods defined in the PrintTicket Validation Checklist of the Print Schema documentation [M2.64]. Following validation, the printing consumer MUST properly interpret the print settings according to the following rules for merging two PrintTicket parts [M2.65].

Print settings are expressed by scoped Print Schema elements. Elements may interact between different levels in the PrintTicket hierarchy. Elements that interact between levels MUST be specified at the root of each level ticket [M2.59]. A keyword merge conflict between PrintTicket settings is defined as the same root-level Print Schema element denoted by the same **name** attribute value appearing in multiple level tickets. There are two options for interactions:

~~1.~~3. If there is no merge conflict, a prefix-scoped element MUST be pushed or inherited, from a more general ticket to a more specific ticket [M2.66]. This case is isomorphic to the case where both tickets contain an identical element.

~~2.~~4. If there is a merge conflict, the setting from the most specific ticket precedence [M2.67]. That is, a page-scoped setting in a page-level PrintTicket overwrites an identical page-scoped setting in a document-level or job-level PrintTicket. Likewise, a document-scoped setting in a document-level PrintTicket overwrites an identical document-scoped setting in a job-level PrintTicket.

To determine the print settings in the XPS Document the following algorithm should be applied:

1. Validate the job-level PrintTicket associated with the fixed document sequence by merging and validating the PrintTicket with the default PrintTicket for the print consumer. The default PrintTicket represents the default configuration of the print consumer's state; in the case of a print consumer, the state is the current device and print driver configuration. Call the resulting ticket the "validated job-level PrintTicket." If no job-level PrintTicket is supplied, use the default PrintTicket.
2. For each FixedDocument*X* referenced by the fixed document sequence, perform the following steps:
   a. Merge and validate the document-level PrintTicket associated with FixedDocument*X* with the validated job-level PrintTicket from Step 1.
   b. Call the resulting ticket the "validated Document*X*-level PrintTicket." If no document-level PrintTicket is supplied for the current fixed document, the validated job-level PrintTicket from Step 1 should be used.
3. For each FixedPage*Y* referenced by each FixedDocument*X* in the fixed document sequence, perform the following steps:
   a. Merge and validate the page-level PrintTicket associated with FixedPage*Y* with the validated Document*X*-level PrintTicket from Step 2.
   b. Call the resulting ticket the "validated Page*XY*-level PrintTicket." If no page-level PrintTicket is supplied for the current fixed page, the validated Document*X*-level PrintTicket from Step 2 should be used.

9.1.10 SignatureDefinitions Part

Producers MAY add digital signature requests and instructions to an XPS Document in the form of signature definitions [O2.23]. A producer MAY sign against an existing signature definition to provide additional signature information [O2.24]. A recipient of the document MAY also sign the XPS Document against a signature definition [O2.25]. (This is referred to as "co-signing.")

Digital signature definitions are stored in a SignatureDefinitions part. A FixedDocument part refers to a SignatureDefinitions part using a relationship of the SignatureDefinitions type. For more information, see H, "Standard Namespaces and Content Types."

The SignatureDefinitions part is OPTIONAL [O2.6]. Signature definitions MUST conform to the Signature Definitions schema as defined in A [M2.72].

For more information on digital signature support in XPS Documents, see 17, "XPS Document Package Features."

9.1.11 DocumentStructure Part

Explicitly authored document structure information is stored in the DocumentStructure part. This part contains the document outline and defines the framework for every element in fixed pages in terms of semantic blocks called *stories*. Stories are split into StoryFragments parts, which contain content structure markup that defines semantic blocks such as paragraphs and tables. For more information, see 16, "Document Structure and Interactivity."

Document structure markup contains a root <DocumentStructure> element. See 16, “Document Structure and Interactivity,” for markup details. The <DocumentStructure> element uses the Document Structure namespace specified in §H.1, “XML Namespace URIs,” on page 405.

The DocumentStructure part is referenced by relationship from the FixedDocument part, with the relationship type as specified in H, “Standard Namespaces and Content Types.” The content type of the DocumentStructure part is also specified in H.

Consumers MAY provide an algorithmic construction of the structure of an XPS Document based on a page-layout analysis [O2.27], but they MUST NOT use such a method to derive structure for any part of the XPS Document included in the DocumentStructure part [M2.68]. A consumer capable of calculating reading order from the layout of the document MUST use the reading order specified in the DocumentStructure part, even though the derived order may be perceived as preferable to the specified order [M2.68].

9.1.12 StoryFragments Part

The StoryFragments part contains content structure markup (such as tables and paragraphs) associated with a single fixed page.

StoryFragments part markup contains a root <StoryFragments> element. See 16, “Document Structure and Interactivity,” for markup details. The <StoryFragments> element uses the Document Structure namespace specified in §H.1, “XML Namespace URIs,” on page 405.

The StoryFragments part is referenced by relationship from its associated FixedPage part. The content type of the StoryFragments part is specified in §H.2, “Content Types,” on page 406.

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 with the extension ".fdseq". [*Example*: "/FixedDocSeq.fdseq" *end example*] [S2.24].

Comment [rcj7]: Is a comma missing here? That is, should there be only 1 segmant, and that segment's has an extension of .fdseq? Or can there me mor ethan 1 segment, but only 1 with that extension?

- 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/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/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/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/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.

9.3.1 Support for Versioning and Extensibility

XPS Document markup has been designed in anticipation of the evolution of this specification. It also allows third parties to extend the markup. XPS Document markup incorporates the Markup Compatibility specification incorporated by the OPC specification.

The following parts MAY include elements and attributes defined in the Markup Compatibility specification [O2.29]:

- DocumentStructure
- FixedDocument
- FixedDocumentSequence
- FixedPage
- Relationships
- Remote Resource Dictionary
- SignatureDefinitions
- StoryFragments

Consumers of these parts MUST support the Markup Compatibility specification [M2.69]. Before attempting to validate one of these parts against a schema, processors MUST remove all markup compatibility elements and attributes and all ignorable elements and attributes not defined in the expected version of XPS Document markup [M2.69].

[*Note*: Markup compatibility elements and attributes that appear in one XPS Document part do not carry through to a second part via an inline URI reference in the XML markup. Likewise the markup compatibility mechanisms do not carry through from part to part via relationship. *end note*]

9.3.2 XML Usage

All XML content of the parts defined in this specification MUST conform to the following validation rules:

1. XML content MUST be encoded using either UTF-8 or UTF-16. If any such part includes an encoding declaration (as defined in §4.3.3 of the XML specification), that declaration MUST NOT name any encoding other than UTF-8 or UTF-16 [M2.70].

2. The XML 1.0 specification allows for the usage of Data Type Definitions (DTDs), which enable Denial of Service attacks, typically through the use of an internal entity expansion technique. As mitigation for this potential threat, DTD content MUST NOT be used in the XML markup defined in this specification, and consumers MUST treat the presence of DTD content as an error [M2.71].
3. If the XML content contains the Markup Compatibility namespace, as described in the Markup Compatibility specification, it MUST be processed to remove Markup Compatibility elements and attributes, ignorable namespace declarations, and ignored elements and attributes before applying further validation rules below [M2.69].
4. XML content MUST be valid against the corresponding XSD schema defined in this specification. In particular, the XML content MUST NOT contain elements or attributes drawn from namespaces that are not explicitly defined in the corresponding XSD unless the XSD allows elements or attributes drawn from any namespace to be present in particular locations in the XML markup [M2.72].
5. XML content MUST NOT contain elements or attributes drawn from "xml" or "xsi" namespaces unless they are explicitly defined in the XSD schema or by other means in the specification [M2.73].

9.3.3 Markup Model

XPS Document markup is an XML-based markup language that uses elements, attributes, and namespaces. The schema for XPS Document markup includes only elements and their attributes, comments, and whitespace. Arbitrary character data intermingled in the markup is not allowed.

Fixed page markup is expressed using elements and attributes and is based on a higher-level abstract model of contents and properties. Some fixed page elements can hold "contents," which are expressed as child elements. Properties may be expressed either as attributes or child elements.

XPS Document markup also uses resources and resource dictionaries, which allow elements to share property values.

9.3.3.1 Namespaces

The following XML namespaces are defined for use in XPS Document markup:

- The XPS Document namespace, the principal namespace used for elements and attributes in fixed page markup. For more information, see H, "Standard Namespaces and Content Types."
- The Resource Dictionary Key namespace, which allows certain XPS Document elements to be included in a resource dictionary, as described in §14.2, "Resources and Resource References," on page 172.
- The Markup Compatibility namespace, which supports the Markup Compatibility specification as defined in the OPC specification.

9.3.3.2 Properties

A *property* is a characteristic of an element. XPS Document property values can be expressed either as property attributes or property elements. *Property values* may be stored in a resource dictionary and referenced by an attribute that uses a special syntax to express its value. For more information, see §14.2, "Resources and Resource References," on page 172.

Properties MUST NOT be set more than once, regardless of the syntax used to specify the value [M2.74]. 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 [M2.74].

Some properties are common to several fixed page elements. For more information, see 14, "Common Properties."

9.3.3.2.1 Composable Property Values

Some fixed page properties are composable, meaning that the page marking effect is determined by combining the property value of a given element with that of its parent and ancestor elements. [*Example*: A <Path> element with an **Opacity** value of 0.5 nested inside a <Canvas> element with an **Opacity** value of 0.5 results in an effective 25% opacity of the <Path> element when rendered. *end example*]

The coordinate space used to render page marking elements is also composable. By default, elements are rendered in a coordinate space with units of 1/96". The *effective coordinate space* for a particular element is created by sequentially applying each parent and ancestor element's affine matrix transformation, specified with the Transform or RenderTransform properties, from outermost to innermost, including the element's own affine matrix transformation.

For more information, see §18.4.1, "Pre-Multiplied Alpha and Superluminous Colors," on page 281.

9.3.3.2.2 Property Attribute Syntax

Some property values can be expressed using simple XML attribute syntax, that is, with a text string. The value of properties used to describe geometries may be expressed using an abbreviated syntax. For more information, see §11.2.3, "Abbreviated Geometry Syntax," on page 90.

Example 9–3. Property attribute syntax

The following syntax can be used to specify the color of a brush:

```
<!-- Property Attribute Syntax -->
<SolidColorBrush Color="#FF0000" />
```

end example]

9.3.3.2.3 Property Element Syntax

Some property values can also be expressed using a child element to describe the property value. These property elements are included to enable usage of the Markup Compatibility mechanisms described in the Markup Compatibility specification. The element name is derived from a combination of a parent element name and the property name, separated by a dot (.) character.

The order of child property elements is significant: they MUST occur before any contents of the parent element and they MUST appear in the sequence specified in the schema [M2.72].

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.

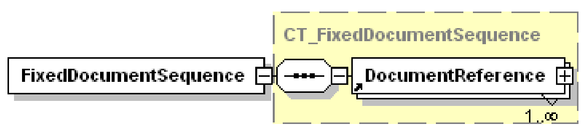
xml:lang is REQUIRED for <FixedPage> elements and MAY be used with <Canvas>, <Path>, and <Glyphs> elements; it is not valid on any other fixed page markup element [M2.72]. **xml:lang** is also REQUIRED for the <DocumentOutline> element for document structure and OPTIONAL for the <OutlineEntry> element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

10. Documents

XPS Documents contain a root fixed document sequence that binds a collection of fixed documents which, in turn, bind a collection of fixed pages. All page markings are specified with <Glyphs> or <Path> elements on the fixed page. These elements can be grouped within one or more <Canvas> elements. Page markings are positioned by real-number coordinates in the coordinate space of the fixed page. The coordinate space can be altered by applying a render transformation.

10.1 <FixedDocumentSequence> Element

element **FixedDocumentSequence**

diagram	
annotation	Specifies a sequence of fixed documents.

The <FixedDocumentSequence> element contains one or more <DocumentReference> elements. The order of <DocumentReference> elements MUST match the order of the documents in the fixed document sequence [M3.1].

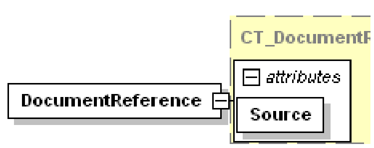
Example 10–1. <FixedDocumentSequence> usage

```
<FixedDocumentSequence xmlns="http://schemas.microsoft.com/xps/2005/06">
   <DocumentReference Source="Documents/1/FixedDocument.fdoc" />
   <DocumentReference Source="Documents/2/FixedDocument.fdoc" />
</FixedDocumentSequence>
```

end example]

10.1.1 <DocumentReference> Element

element **DocumentReference**

diagram	
attributes	see below

Name	Type	Use	Default	Fixed	Annotation
Source	xs:anyURI	required			Specifies the URI of the fixed document content. The specified URI MUST refer to a FixedDocument part within the XPS Document [M3.2].

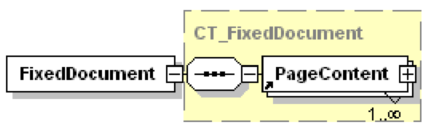
annotation	Contains a reference to a FixedDocument part.

The <DocumentReference> element specifies a FixedDocument part as a URI in the **Source** attribute. Producers MUST NOT produce a document with multiple <DocumentReference> elements that reference the same fixed document [M3.3].

10.2 <FixedDocument> Element

element **FixedDocument**

diagram



annotation	Binds an ordered sequence of fixed pages together into a single multi-page document.

The <FixedDocument> element contains one or more <PageContent> elements. The order of <PageContent> elements MUST match the order of the pages in the document [M3.4].

Example 10–2. <FixedDocument> usage

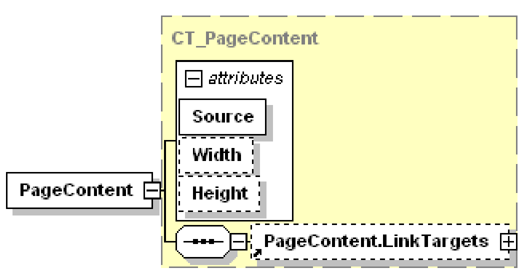
```
<FixedDocument xmlns="http://schemas.microsoft.com/xps/2005/06">
   <PageContent Source="Pages/1.fpage" />
   <PageContent Source="Pages/2.fpage" />
</FixedDocument>
```

end example]

10.2.1 <PageContent> Element

element **PageContent**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
Source	xs:anyURI	required			Specifies a URI that refers to the page content, held in a distinct part within the package. The content identified MUST be a FixedPage part

						within the XPS Document [M3.5].
	Width	ST_GEOne				Typical width of pages contained in the page content.
	Height	ST_GEOne				Typical height of pages contained in the page content.
annotation	Defines a reference from a fixed document to a part that contains a <FixedPage> element.					

Each <PageContent> element refers to the source of the content for a single page. The number of pages in the document can be determined by counting the number of <PageContent> elements.

The <PageContent> element has a single required attribute, **Source**, which refers to a FixedPage part. It may optionally include advisory **Height** and **Width** attributes to indicate the size of a single page. (The authoritative height and width are specified by the fixed page.) The **Height** and **Width** attribute values allow consumers such as viewers to make initial visual layout estimates quickly, without loading and parsing all of the individual fixed pages. These consumers then update the page dimensions when the fixed page is loaded, if they differ.

The <PageContent> element has one allowable child element, <PageContent.LinkTargets>, and it MUST NOT contain more than a single child element [M2.72].

Producers MUST NOT produce markup where a <PageContent> element references the same fixed page referenced by any other <PageContent> element in the entire XPS Document, even in other fixed documents within the fixed payload [M3.6].

Example 10–3. <PageContent> usage

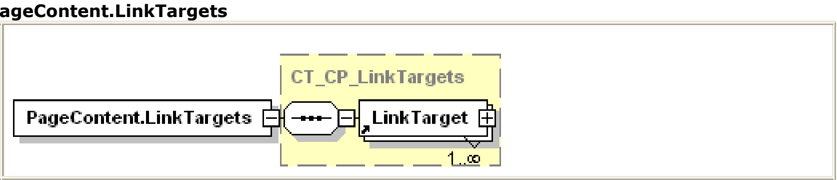
```
<FixedDocument xmlns="http://schemas.microsoft.com/xps/2005/06">
   <PageContent Source="Pages/1.fpage" Height="1056" Width="816" />
   <PageContent Source="Pages/2.fpage" Height="1056" Width="816" />
</FixedDocument>
```

end example]

10.2.2 <PageContent.LinkTargets> Element

element **PageContent.LinkTargets**

diagram



annotation: Contains a collection of <LinkTarget> elements, each of which is addressable via hyperlink.

The <PageContent.LinkTargets> element defines the list of link targets that specify each named element on the page that may be addressed by hyperlink.

Example 10–4. <PageContent.LinkTargets> usage

In the following markup, `Pages/2.fpage` contains two <LinkTarget> elements with **Name** attribute values of `Anchor1` and `Anchor2`:

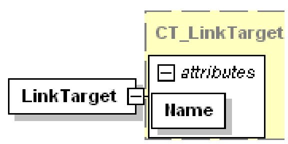
```
<FixedDocument xmlns="http://schemas.microsoft.com/xps/2005/06">
   <PageContent Source="Pages/1.fpage" Height="1056" Width="816" />
   <PageContent Source="Pages/2.fpage" Height="1056" Width="816">
      <PageContent.LinkTargets>
         <LinkTarget Name="Anchor1" />
         <LinkTarget Name="Anchor2" />
      </PageContent.LinkTargets>
   </PageContent>
</FixedDocument>
```

end example]

10.2.3 <LinkTarget> Element

element **LinkTarget**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
Name	ST_Name	required			Contains a string value that identifies the current element as a named, addressable point in the document for the purpose of hyperlinking.

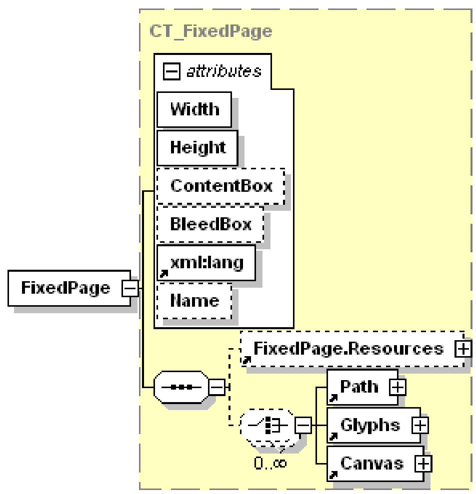
annotation: Specifies an addressable point on the page.

The <LinkTarget> element specifies a **Name** attribute, which corresponds to a named location within the fixed page specified by its parent <PageContent> element. By encapsulating this information in the fixed document, consumers do not need to load every FixedPage part to determine if a particular **Name** value exists in the document. For more information, see §16.2, “Hyperlinks,” on page 243.

10.3 <FixedPage> Element

element **FixedPage**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
Width	ST_GEOne	required			Width of the page, expressed as a real number in units of the effective coordinate space.
Height	ST_GEOne	required			Height of the page, expressed as a real number in units of the effective coordinate space.
ContentBox	ST_ContentBox				Specifies the area of the page containing imageable content that is to be fit within the imageable area when printing or viewing. Contains a list of four coordinate values (ContentOriginX, ContentOriginY, ContentWidth, ContentHeight), expressed as comma-separated real numbers. Specifying a value is RECOMMENDED [S3.1]. If omitted, the default value is (0,0,Width,Height).
BleedBox	ST_BleedBox				Specifies the area including crop marks that extends outside of the physical page. Contains a list of four coordinate values (BleedOriginX, BleedOriginY,

						BleedWidth, BleedHeight), expressed as comma-separated real numbers. If omitted, the default value is (0,0,Width,Height).
	xml:lang		required			Specifies the default language used for the current element and for any child or descendant elements. The language is specified according to RFC 3066.
	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.
annotation	Contains markup that describes the rendering of a single page of content.					

The <FixedPage> element contains the contents of a page and is the root element of a FixedPage part. The fixed page contains the elements that together form the basis for all markings rendered on the page: <Paths>, <Glyphs>, and the optional <Canvas> grouping element.

The fixed page MUST specify a height, width, and default language [M2.72].

The coordinate space of the fixed page is composable, meaning that the marking effects of its child and descendant elements are affected by the coordinate space of the fixed page.

Example 10–5. Fixed page markup

```
<FixedPage Height="1056" Width="816" xml:lang="en-US"
   xmlns="http://schemas.microsoft.com/xps/2005/06">
   <Glyphs
      OriginX="96"
      OriginY="96"
      UnicodeString="This is Page 1!"
      FontUri="../Resources/Fonts/Times.TTF"
      FontRenderingEmSize="16" />
</FixedPage>
```

end example]

10.3.1 BleedBox Attribute

The **BleedBox** attribute defines the area (inclusive of crop marks) that extends outside of the physical page. The bleed box is expressed as four comma-separated, real-number coordinate values that correspond to BleedOriginX, BleedOriginY, BleedWidth, BleedHeight. These values are specified in units of 1/96".

Bleed boxes that do not satisfy the following conditions are invalid and SHOULD be ignored in favor of the default bleed box [S3.2]:

- The **BleedBox** BleedOriginX value MUST be less than or equal to 0 [M3.7].
- The **BleedBox** BleedOriginY value MUST be less than or equal to 0 [M3.8].

- The **BleedBox** BleedWidth value MUST be greater than or equal to the fixed page **Width** attribute value plus the absolute value of the **Bleedbox** BleedOriginX value [M3.9].
- The **BleedBox** BleedHeight value MUST be greater than or equal to the fixed page **Height** attribute value plus the absolute value of the **BleedBox** BleedOriginY value [M3.10].

If the **BleedBox** attribute is omitted, the default value is "0,0,*Width*,*Height*".

10.3.2 ContentBox Attribute

The **ContentBox** attribute specifies the area of the page that contains imageable content that must fit in the imageable area when printing or viewing. Specifying this attribute is RECOMMENDED [S3.1]. The content box is expressed as four comma-separated, real-number coordinate values that correspond to ContentOriginX, ContentOriginY, ContentWidth, ContentHeight. These values are specified in units of 1/96".

Content boxes that do not satisfy the following conditions are invalid and SHOULD be ignored in favor of the default content box [S3.3]:

- The **ContentBox** ContentOriginX value MUST be greater than or equal to 0 and less than the fixed page **Width** attribute value [M3.11].
- The **ContentBox** ContentOriginY value MUST be greater than or equal to 0 and less than the fixed page **Height** attribute value [M3.12].
- The **ContentBox** ContentWidth value MUST be less than or equal to the difference between the fixed page **Width** attribute value and the **ContentBox** ContentOriginX value [M3.13].
- The **ContentBox** ContentHeight value MUST be less than or equal to the difference between the fixed page **Height** attribute value and the **ContentBox** ContentOriginY value [M3.14].

If the **ContentBox** attribute is omitted, the default value is "0,0,*Width*,*Height*".

10.3.3 Page Size Terminology

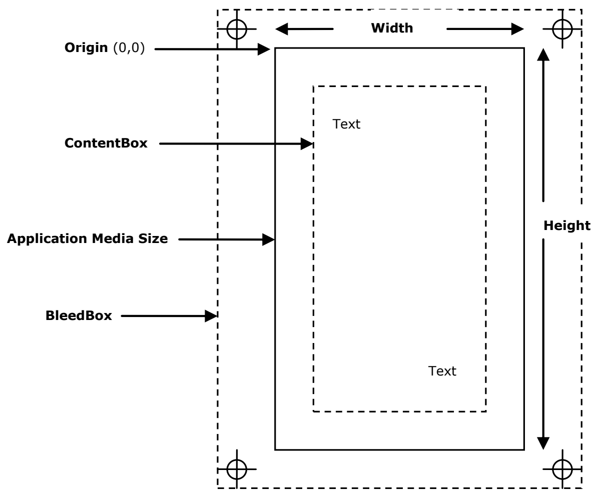
The following terminology is used to describe page sizes:

- The *producer media size* represents the media size that is used by the producer for layout purposes. This size is described by the **Height** and **Width** attributes.
- The *producer bleed size* represents the overflow or "bleed" box used by the producer for registration and layout. This size is described by the **BleedBox** attribute. If the **BleedBox** attribute is not present, then the producer bleed size is defined as the producer media size.
- The *producer content size* represents the content bounding box specified by the producer. This size is described by the **ContentBox** attribute. If the **ContentBox** attribute is not present, then the producer content size is defined as the producer media size.
- The *physical media size* represents the physical media on which the content will be printed. This size is described by the PageMediaSize keyword in the

Print Schema. The PageMediaSize is page-orientation-agnostic. This means that the representation of physical media size is defined by two dimensions, PageMediaSizeWidth and PageMediaSizeHeight. For more information, see the Print Schema specification.

- The *physical imageable size* represents the area that is printable by a specific device. This size is described by the PageImageableSize keyword in the Print Schema. The PageImageableSize is relative to the physical media size (PageMediaSize keyword) and the orientation (PageOrientation keyword). For more information, see the Print Schema specification.

Figure 10–1. Page regions



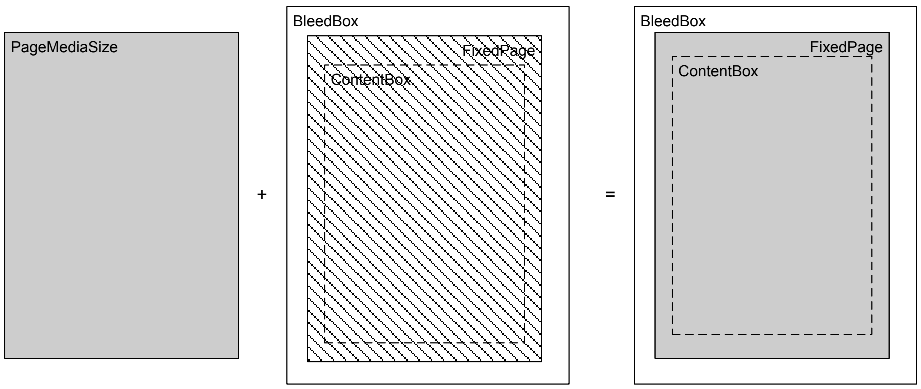
10.3.4 Media Orientation and Scaling

When rendering a fixed page for printing, consumers MUST be aware of the interaction between the fixed page markup and the PrintTicket settings [M3.15]. The interaction for media scaling is governed by the PageMediaSize, PageImageableSize, PageScaling, and PageOrientation Print Schema keywords.

For media orientation, the PageMediaSize, PageOrientation and the width and height of the <FixedPage> element determine the rendering of the fixed page. In the absence of media scaling, the fixed page content is imaged directly to the physical media with the origin of the fixed page aligned with the origin of the physical media size. Any fixed page content that extends beyond the dimension of the physical media size SHOULD be clipped [S3.4].

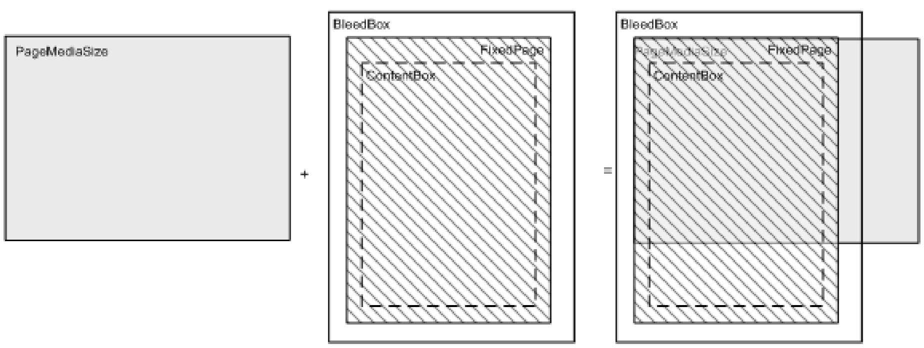
[*Example*: Consider a fixed page with a width of 816 (8.5") and a height of 1056 (11"). If the PrintTicket specifies the PageMediaSize value as Letter and the PageOrientation value as Portrait then no clipping occurs, as shown below.

Figure 10–2. Matching PrintTicket and fixed page size and orientation



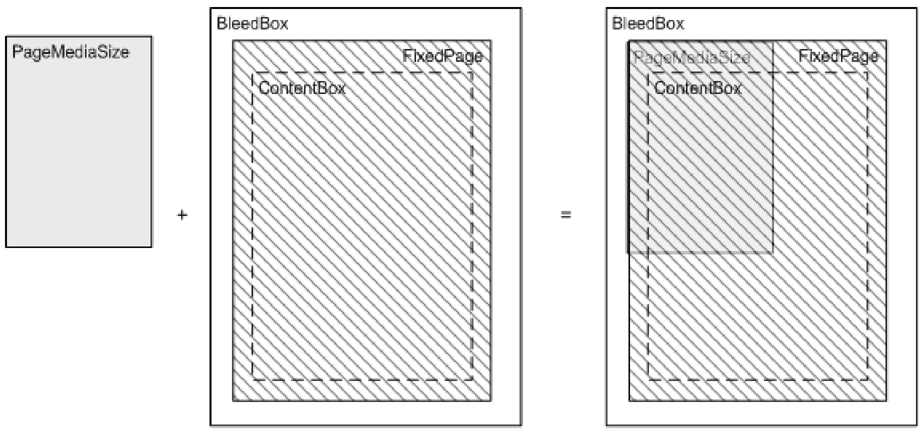
If the PrintTicket specifies the PageMediaSize value as Letter and the PageOrientation value as Landscape the bottom of the fixed page content is clipped, as shown below.

Figure 10–3. Matching PrintTicket and fixed page size with differing orientation



Finally, if the PrintTicket specifies the PageMediaSize value as 4x6 and the PageOrientation value as Portrait, the bottom and right portions of the fixed page content are clipped, as shown below.

Figure 10–4. Matching PrintTicket and fixed page orientation with differing size



end example]

[*Note*: Fixed pages intended for landscape-orientation printing must be produced differently than those intended for portrait-orientation printing; their **Width** attribute holds the larger value, their **Height** attribute holds the smaller value.

The PageOrientation setting in the PrintTicket does not rotate the fixed page, but instead determines how the PageMediaSize dimensions relate to the PageImageableSize dimensions reported by the device.

As a consequence, in order to print a fixed page that has been produced with a landscape orientation, either the PrintTicket would specify PageMediaSize and PageOrientation values such that the dimensions of the fixed page match the PageImageableSize reported by the device, or it would specify an appropriate PageScaling option. *end note*]

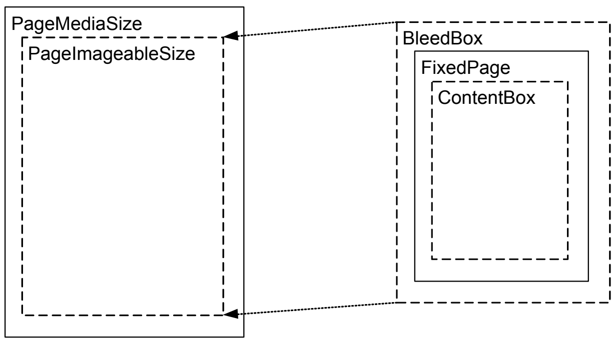
For media scaling, the following page scaling options determine the rendering of the fixed page:

- FitApplicationBleedSizeToPageImageableSize
- FitApplicationContentSizeToPageImageableSize
- FitApplicationMediaSizeToPageImageableSize
- FitApplicationMediaSizeToPageMediaSize

This is not an exhaustive list; for more information, see the Print Schema specification.

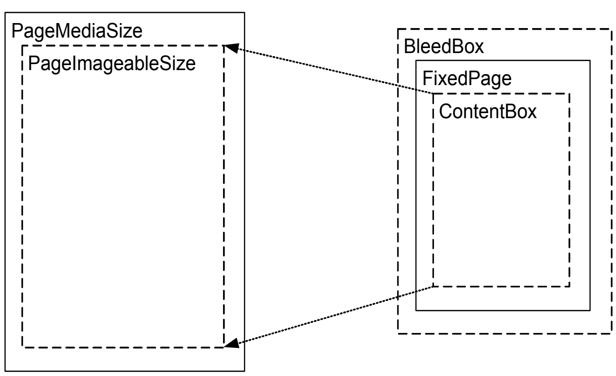
10.3.4.1 FitApplicationBleedSizeToPageImageableSize

Consumers MUST scale the bleed box (producer bleed size) to the PageImageableSize, preserving the aspect ratio [M3.16]. See the Print Schema PageScaling definition.



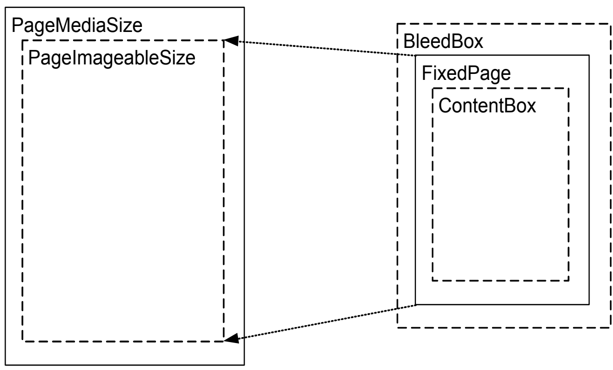
10.3.4.2 FitApplicationContentSizeToPageImageableSize

Consumers MUST scale the content box (producer content size) to the PageImageableSize, preserving the aspect ratio [M3.17]. See the Print Schema PageScaling definition.



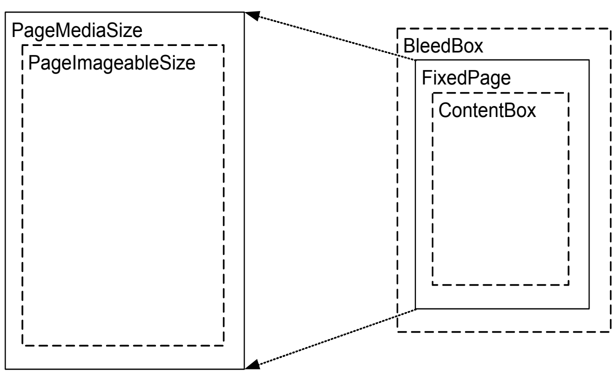
10.3.4.3 FitApplicationMediaSizeToPageImageableSize

Consumers MUST scale the height and width (producer media size) to the PageImageableSize, preserving the aspect ratio [M3.18]. See the Print Schema PageScaling definition.



10.3.4.4 FitApplicationMediaSizeToPageMediaSize

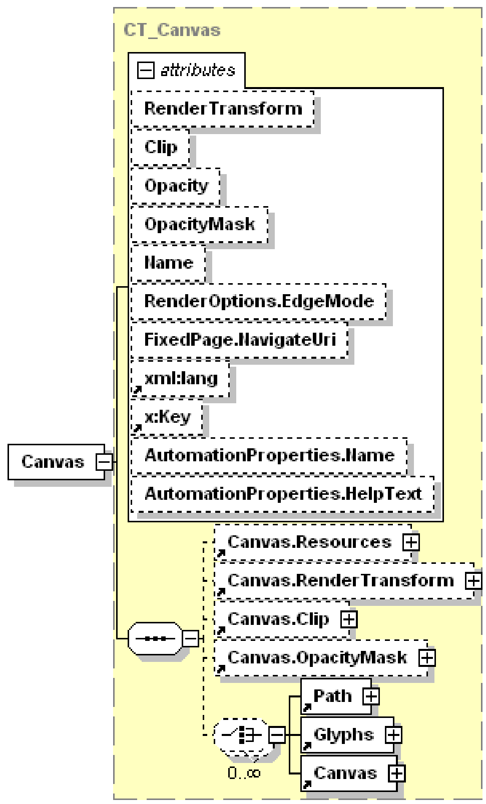
Consumers MUST scale the height and width (producer media size) to the PageMediaSize, preserving the aspect ratio [M3.19]. See the Print Schema PageScaling definition.



10.4 <Canvas> Element

element **Canvas**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
RenderTransform	ST_RscRefMatrix				Establishes a new coordinate frame for the child and descendant elements of the canvas, such as another canvas. Also affects clip and opacity mask.
Clip	ST_RscRefAbbrGeomF				Limits the rendered region of

						the element.
	Opacity	ST_ZeroOne		1.0		Defines the uniform transparency of the canvas. 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 canvas 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.
	RenderOptions.EdgeMode	ST_EdgeMode				Controls how edges of paths within the canvas are rendered. The only valid value is Aliased. Omitting this attribute causes the edges to be rendered in the consumer's default manner.
	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 and for any child or descendant elements. 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 [M3.20].
	AutomationProperties.Name	xs:string				A brief description of the <Canvas> contents for accessibility purposes, particularly if filled with a set of vector graphics and text elements intended to comprise a single vector graphic.
	AutomationProperties.HelpText	xs:string				A detailed description of the <Canvas> contents for accessibility purposes, particularly if filled with a set of

						graphics and text elements intended to comprise a single vector graphic.
annotation	Groups <FixedPage> descendant elements together.					

The <Canvas> element groups elements together. [*Example*: <Glyphs> and <Path> elements can be grouped in a canvas in order to be identified as a unit (as a hyperlink destination) or to apply a composed property value to each child and ancestor element. *end example*]

Some properties of the <Canvas> element are composable and affect the rendering of child elements. This includes the coordinate space of the canvas. For details, see 14, “Common Properties.”

The RenderOptions.EdgeMode property can be set on the <Canvas> element to instruct anti-aliasing consumers to render the contents of the <Canvas> and all child and descendant elements without performing anti-aliasing, including child brushes and their contents as well as contents included via resource dictionary references.

Example 10–6. Canvas composition

The following markup describes a path that provides the background. On top of this is rendered a canvas with the composable Opacity and RenderTransform properties specified.

The path inside the canvas has the same path geometry as the background path, but since it is composing the <Canvas> element’s RenderTransform property, it is rendered differently. The path is partially transparent due to the composable Opacity property of the parent <Canvas> element.

The <Glyphs> element inside the canvas specifies its own RenderTransform property. This property is composed with the <Canvas> element’s RenderTransform property, such that the coordinate space of the <Glyphs> element is transformed within the context of the coordinate space transformed by the <Canvas> element.

```
<Path>
   <Path.Fill>
      <SolidColorBrush Color="#808080" />
   </Path.Fill>
   <Path.Data>
      <PathGeometry>
         <PathFigure StartPoint="0,0" IsClosed="true">
            <PolyLineSegment Points="200,0 200,100 0,100 0,0" />
         </PathFigure>
      </PathGeometry>
   </Path.Data>
</Path>

<Canvas Opacity="0.5" RenderTransform="0.75,0,0,0.75,25,46">
   <Path>
      <Path.Fill>
         <SolidColorBrush Color="#0000FF" />
      </Path.Fill>
      <Path.Data>
```

```
            <PathGeometry>
               <PathFigure StartPoint="0,0" IsClosed="true">
                  <PolyLineSegment Points="200,0 200,100 0,100 0,0" />
               </PathFigure>
            </PathGeometry>
         </Path.Data>
      </Path>
      <Glyphs
         FontUri="../Resources/Fonts/times.ttf"
         OriginX="1"
         OriginY="100"
         UnicodeString="EXAMPLE"
         FontRenderingEmSize="42"
         RenderTransform="1.0,0,0,2.0,0,-100">
         <Glyphs.Fill>
            <SolidColorBrush Color="#FFFFFF" />
         </Glyphs.Fill>
      </Glyphs>
   </Canvas>
```

This markup is rendered as follows:

end example]

10.5 <Path> Element

The <Path> element specifies a geometry that can be filled with a brush. For more information, see §11.1, "<Path> Element," on page 68.

10.6 <Glyphs> Element

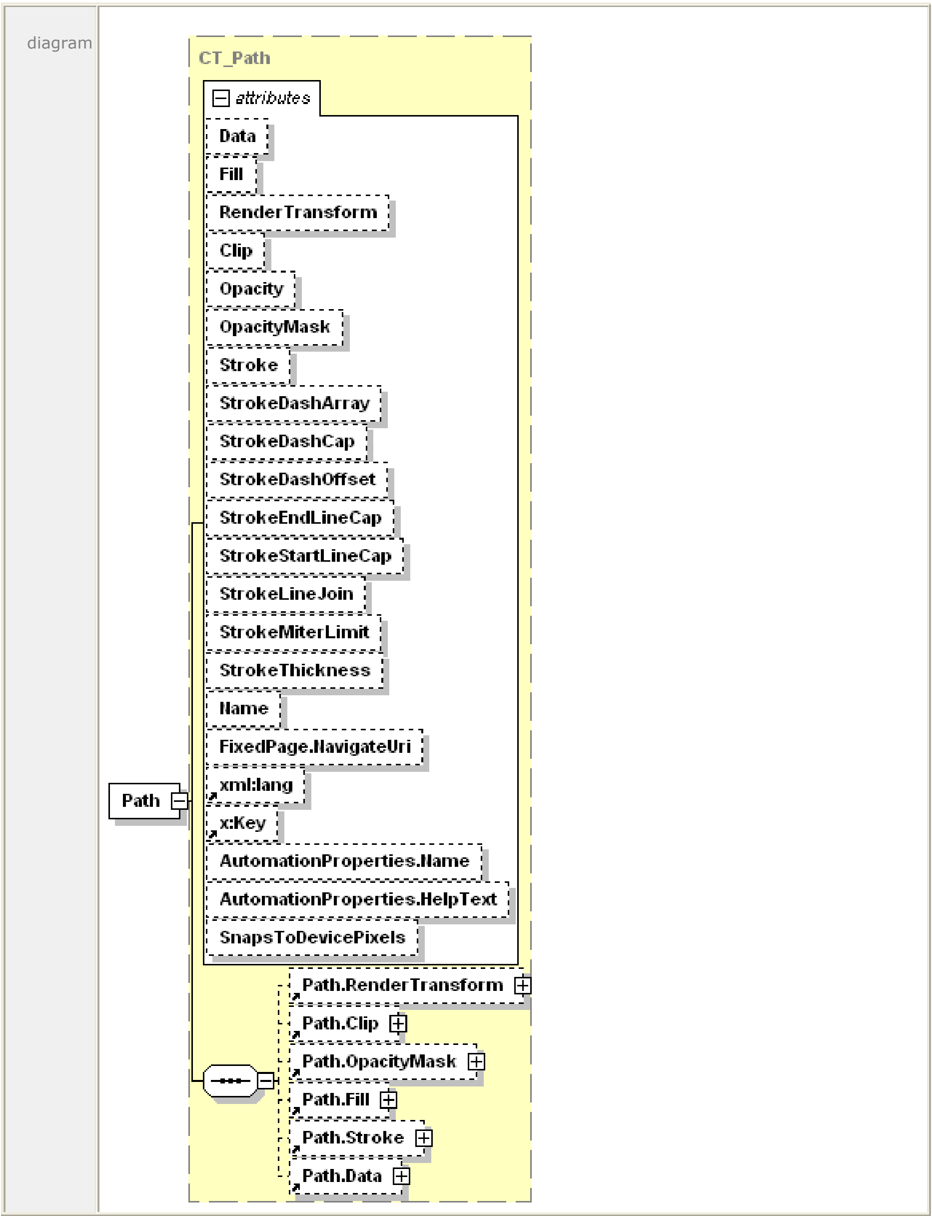
The <Glyphs> element is used to represent a run of uniformly-formatted text from a single font. The <Glyphs> element provides information for accurate rendering and supports search and selection features in XPS Document consumers. For more information, see §12.1, "<Glyphs> Element," on page 98.

11. Graphics

Vector graphics are created using the <Path> element. A full set of properties is available to describe the visual characteristics of the graphic. These characteristics include the fill, opacity, clipping, rendering transformation, and various stroke details including thickness, fill, line join style, line miter limit, line cap style, dash style, and dash cap style. The description of the geometric area of the path (the geometry) is described by the Data property. Raster images are included in fixed page markup by specifying a <Path> element filled with an <ImageBrush>.

11.1 <Path> Element

element **Path**



attributes	Name	Type	Use	Default	Fixed	Annotation
	Data	ST_RscRefAbbrGeomF				Describes the geometry of the path.
	Fill	ST_RscRefColor				Describes the brush used to paint the geometry specified by the Data property of the path.
	RenderTransform	ST_RscRefMatrix				Establishes a new coordinate frame for all attributes of the path and for all child elements of the path, such as the geometry defined by the <Path.Data> property element.
	Clip	ST_RscRefAbbrGeomF				Limits the rendered region of the element.
	Opacity	ST_ZeroOne		1.0		Defines the uniform transparency of the path 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 path in the same fashion as the Opacity attribute, but allowing different alpha values for different areas of the element.
	Stroke	ST_RscRefColor				Specifes the brush used to draw the stroke.
	StrokeDashArray	ST_EvenArrayPos				Specifies the length of dashes and gaps of the outline stroke. These values are specified as multiples of the stroke thickness as a space-separated list with an even number of non-negative values. When a stroke is drawn, the dashes and gaps specified by these values are repeated to cover the length of the stroke. If this attribute is omitted, the stroke is drawn solid, without any gaps.
	StrokeDashCap	ST_DashCap		Flat		Specifies how the ends of each dash are drawn. Valid values are Flat, Round, Square, and Triangle.
	StrokeDashOffset	ST_Double		0.0		Adjusts the start point for repeating the dash array

						pattern. If this value is omitted, the dash array aligns with the origin of the stroke. Values are specified as multiples of the stroke thickness.
	StrokeEndLineCap	ST_LineCap		Flat		Defines the shape of the end of the last dash in a stroke. Valid values are Flat, Square, Round, and Triangle.
	StrokeStartLineCap	ST_LineCap		Flat		Defines the shape of the beginning of the first dash in a stroke. Valid values are Flat, Square, Round, and Triangle.
	StrokeLineJoin	ST_LineJoin		Miter		Specifies how a stroke is drawn at a corner of a path. Valid values are Miter, Bevel, and Round. If Miter is selected, the value of StrokeMiterLimit is used in drawing the stroke.
	StrokeMiterLimit	ST_GEOne		10.0		The ratio between the maximum miter length and half of the stroke thickness. This value is significant only if the StrokeLineJoin attribute specifies Miter.
	StrokeThickness	ST_GEZero		1.0		Specifies the thickness of a stroke, in units of

						the effective coordinate space (includes the path's render transform). The stroke is drawn on top of the boundary of the geometry specified by the <Path> element's Data property. Half of the StrokeThickness extends outside of the geometry specified by the Data property and the other half extends inside of the geometry.
	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. Can 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 and for any child or descendant elements. 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 [M4.1].
	AutomationProperties.Name	xs:string				A brief description of the <Path> for accessibility purposes, particularly if filled with an <ImageBrush>.
	AutomationProperties.HelpText	xs:string				A detailed description of the <Path> for accessibility purposes, particularly if filled with an <ImageBrush>.
	SnapsToDevicePixels	ST_Boolean				On Anti-aliasing consumers controls if control points snap to the nearest device pixels. Valid values are 'false' and 'true'. Consumers MAY ignore this attribute [O4.1].
annotation	Defines a single graphical effect to be rendered to the page. It paints a geometry with a brush and draws a stroke around it.					

The <Path> element is the sole means of adding vector graphics and images to a fixed page. It defines a single vector graphic to be rendered on a page. Some properties of the <Path> element are composable, meaning that the markings rendered to the page are determined by a combination of the property and all of the like-named properties of its parent and ancestor elements.

The Data property contains a geometric description of the area on which to apply a given effect. This description can take one of two forms: verbose or abbreviated. In the verbose form, the geometry is described in the <Path.Data> property element using the elements described in §11.2, "Geometries and Figures," on page 77. In abbreviated form, it is described using abbreviated syntax in the **Data** attribute. For more information, see §11.2.3, "Abbreviated Geometry Syntax," on page 90.

The <Path.Fill> property element describes the appearance of the area specified by the Data property. It contains a brush (see 13, "Brushes") that is used to fill the described areas. These can include a solid color, an image, a gradient, or a vector drawing pattern.

The <Path.Stroke> property element describes the appearance of the borders of the shape specified by the Data property. It also contains a <Brush> element, which is used to fill the borders according to the stroke properties (such as StrokeThickness). See 18, "Rendering Rules," for detailed rendering rules of strokes, line caps, and dash caps.

If neither Stroke nor Fill properties are specified, the <Path> element has no visible effect.

The transparency of the rendered <Path> element is controlled by the **Opacity** attribute. More complex transparency descriptions may be defined using the **OpacityMask** attribute to control the transparency of the brush described by the Fill property.

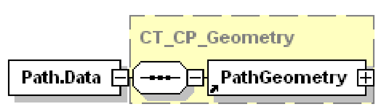
Consumers or viewers that perform anti-aliasing MAY "snap" those control points of the path that are situated on the path bounding box to whole device pixels if the ignorable **SnapsToDevicePixels** attribute is specified as true [O4.1].

Finally, the path may be cropped by specifying a clipping region in the Clip property, which describes the geometric area to be preserved. The remainder is not rendered. See §11.2.1, "Geometries," on page 78 for how geometries are defined.

For details on the Clip, Opacity, OpacityMask, and RenderTransform properties, see 14, "Common Properties."

11.1.1 <Path.Data> Element

element **Path.Data**

diagram	
annotation	Describes the geometry of the path.

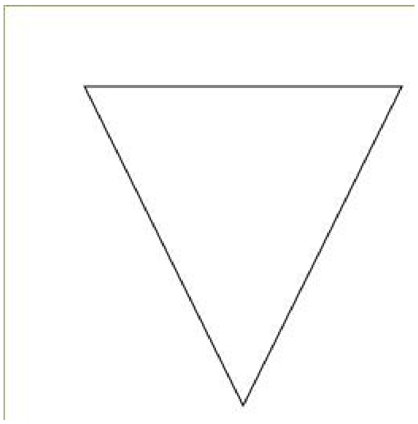
The <Path.Data> property element describes the geometric area of a path. It contains a single geometry.

Example 11–1. <Path.Data> usage

```
<Path Stroke="#000000" StrokeThickness="1">
   <Path.Data>
      <PathGeometry>
```

```
            <PathFigure StartPoint="50,50" IsClosed="true">
               <PolyLineSegment Points="250,50 150,250" />
            </PathFigure>
         </PathGeometry>
      </Path.Data>
   </Path>
```

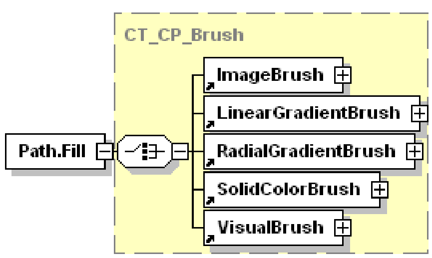
This markup produces the following results:



end example]

11.1.2 <Path.Fill> Element

element **Path.Fill**

diagram	
annotation	Describes the brush used to paint the geometry specified by the Data property of the path.

The <Path.Fill> property element specifies the brush that is used to fill the region described by the Data property. This can be a solid color, an image, a gradient, or a vector drawing pattern.

Example 11–2. <Path.Fill> usage

In the following markup, the geometry is filled with a solid color:

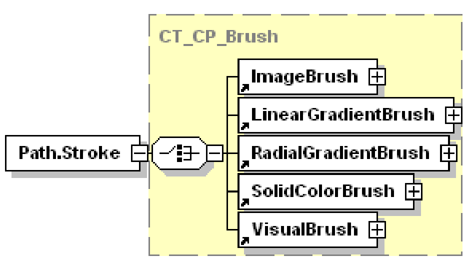
```
<Path>
   <Path.Fill>
      <SolidColorBrush Color="#0000FF" />
   </Path.Fill>
   <Path.Data>
      <PathGeometry>
         <PathFigure StartPoint="10,10" IsClosed="true">
            <PolyLineSegment Points="50,200 100,40 150,200
               200,10 100,105" />
         </PathFigure>
      </PathGeometry>
   </Path.Data>
</Path>
```

This markup produces the following result:

end example]

11.1.3 <Path.Stroke> Element

element **Path.Stroke**

diagram	
annotation	Specifies the brush used to draw the stroke.

The <Path.Stroke> property element describes the border of the path's geometry. <Path.Stroke> contains a brush. Only those segments of the path figure in the

<Path.Data> element that set the **IsStroked** attribute to true (the default value if omitted) are stroked. If **IsClosed** is set to true, an extra segment will be stroked, connecting the last point in the path figure with the first point in the path figure.

The <Path.Stroke> property element is then used to describe the appearance of the borders of the shape defined by the Data property. It also contains a brush, which is used to fill the borders according to the stroke properties (such as StrokeThickness).

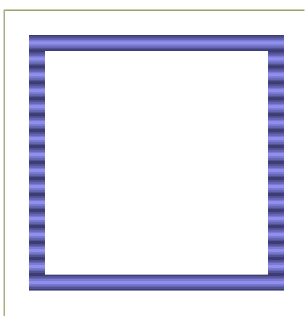
For more information, see §18.6, "Stroke Rendering," on page 288.

Example 11–3. <Path.Stroke> usage

The following <Path.Stroke> element uses a gradient brush to fill the border of a box:

```
<Path StrokeThickness="10" Data="M 20,20 L 170,20 L 170,170 L 20,170 Z">
   <Path.Stroke>
      <LinearGradientBrush MappingMode="Absolute"
         StartPoint="0,0" EndPoint="0,5" SpreadMethod="Reflect">
         <LinearGradientBrush.GradientStops>
            <GradientStop Color="#9999FF" Offset="0.0" />
            <GradientStop Color="#333366" Offset="1.0" />
         </LinearGradientBrush.GradientStops>
      </LinearGradientBrush>
   </Path.Stroke>
</Path>
```

This markup produces the following results:



end example]

11.2 Geometries and Figures

Geometries are used to build visual representations of geometric shapes.

The smallest atomic unit in a geometry is a segment. Segments may be lines or curves. One or more segments are combined into a path figure definition. A path figure is a single shape comprised of continuous segments. One or more path figures collectively define an entire path geometry. A path geometry MAY define the fill algorithm to be used on the component path figures [M2.72].

A single path geometry may be used in the Data property of the <Path> element to describe its overall geometry. A path geometry may also be used in the Clip property of the <Canvas>, <Path>, or <Glyphs> elements to describe a clipping region.

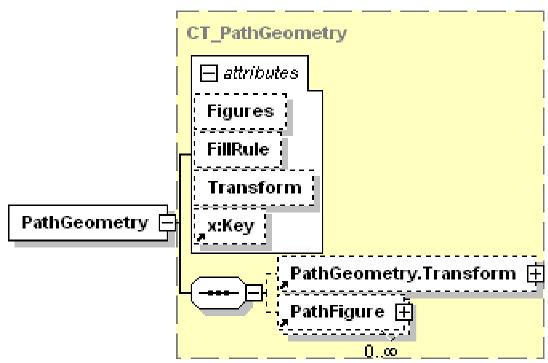
11.2.1 Geometries

A <PathGeometry> element constitutes a complete geometry definition.

11.2.1.1 <PathGeometry> Element

element **PathGeometry**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
Figures	ST_AbbrGeom				Describes the geometry of the path.
FillRule	ST_FillRule		EvenOdd		Specifies how the intersecting areas of geometric shapes are combined to form a region. Valid values are EvenOdd and NonZero.
Transform	ST_RscRefMatrix				Specifies the local matrix transformation that is applied to all child and descendant elements of the path geometry before it is used for filling, clipping, or stroking.
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 [M4.2].

annotation: Contains a set of <PathFigure> elements.

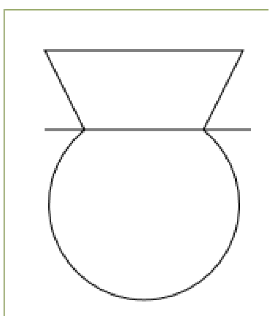
A <PathGeometry> element contains a set of path figures specified either with the **Figures** attribute or with a child <PathFigure> element. Producers MUST NOT specify the path figures of a geometry with both the **Figures** attribute and a child <PathFigure> element [M4.3].

The union of the path figures defines the interior of the path geometry according to the **FillRule** attribute as described in §11.2.1.2, "FillRule Attribute," on page 79.

Example 11–4. <PathGeometry> usage

```
<Path Stroke="#000000">
   <Path.Data>
      <PathGeometry>
         <PathFigure StartPoint="25,75">
            <PolyLineSegment Points="150,75 50,75" />
         </PathFigure>
         <PathFigure StartPoint="50,75" IsClosed="true">
            <ArcSegment
               Size="60,60"
               RotationAngle="0"
               IsLargeArc="true"
               SweepDirection="Counterclockwise"
               Point="125,75" />
         </PathFigure>
         <PathFigure StartPoint="50,75" IsClosed="true">
            <PolyLineSegment Points="25,25 150,25 125,75" />
         </PathFigure>
      </PathGeometry>
   </Path.Data>
</Path>
```

This markup produces the following results:



end example]

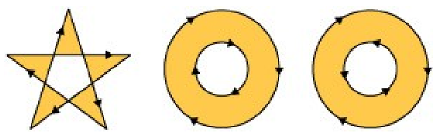
11.2.1.2 FillRule Attribute

The **FillRule** attribute specifies a fill algorithm. The fillable area of a geometry is defined by taking all of the contained path figures and applying the fill algorithm to determine the enclosed area. Fill algorithms determine how the intersecting areas of geometric shapes are combined to form a region.

11.2.1.2.1 EvenOdd Fill Algorithm

This rule determines the "insideness" of a point on the canvas by drawing a ray from the point to infinity in any direction and counting the number of segments from the given shape that the ray crosses. If this number is odd, the point is inside; if it is even, the point is outside. This is the default rule used throughout XPS Document markup.

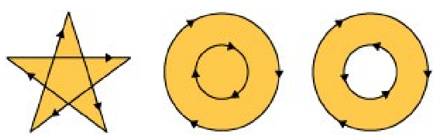
Figure 11–1. Fill using EvenOdd algorithm



11.2.1.2.2 NonZero Fill Algorithm

This rule determines the "insideness" of a point on the canvas by drawing a ray from the point to infinity in any direction and then examining the places where a segment of the shape crosses the ray. Starting with a count of zero, add one each time a segment crosses the ray from left to right and subtract one each time a path segment crosses the ray from right to left. After counting the crossings, if the result is zero then the point is outside the path; otherwise, it is inside.

Figure 11–2. Fill using NonZero algorithm



11.2.1.3 Figures Attribute

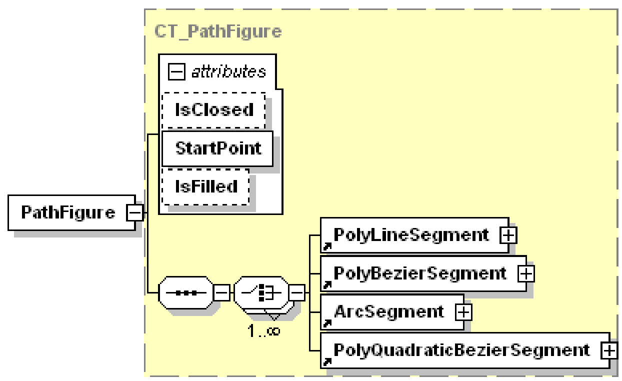
The <PathGeometry> element's **Figures** attribute can be used to describe the path figures the geometry contains using abbreviated syntax (see §11.2.3, "Abbreviated Geometry Syntax," on page 90) with the exception that the FillRule command MUST NOT be used [M2.72].

11.2.2 Figures

11.2.2.1 <PathFigure> Element

element **PathFigure**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
IsClosed	ST_Boolean		false		Specifies whether the path is closed. If set to true, the stroke is drawn "closed," that is, the last point in the last segment of the path figure is connected with the point specified in the StartPoint attribute, otherwise the stroke is drawn "open," and the last point is not connected to the start point. Only applicable if the path figure is used in a <Path> element that specifies a stroke.
StartPoint	ST_Point	required			Specifies the starting point for the first segment of the path figure.
IsFilled	ST_Boolean		true		Specifies whether the path figure is used in computing the area of the containing path geometry. Can be true or false. When set to false, the path figure is considered only for stroking.

annotation

Specifies a set of one or more segment elements defining a closed region.

A <PathFigure> element is composed of a set of one or more line or curve segments. The segment elements define the shape of the path figure. The initial point of the first segment element is specified as the **StartPoint** attribute of the path figure. The last point of each segment element is the first point of the following segment element.

Segment elements are:

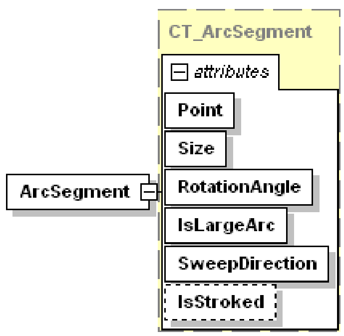
- <ArcSegment>
- <PolyBezierSegment>
- <PolyLineSegment>
- <PolyQuadraticBezierSegment>

Line segments and curve segments SHOULD NOT be specified as zero-length [S4.1]. If they are specified as zero-length, nothing is drawn, even if line caps would normally be visible. For further details and exceptions, see 18, “Rendering Rules.”

11.2.2.2 <ArcSegment> Element

element **ArcSegment**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
Point	ST_Point	required			Specifies the endpoint of the elliptical arc.
Size	ST_PointGE0	required			Specifies the x and y radius of the elliptical arc as an x,y pair.
RotationAngle	ST_Double	required			Indicates how the ellipse is rotated relative to the current coordinate system.
IsLargeArc	ST_Boolean	required			Determines whether the arc is drawn with a sweep of 180 or greater. Can be true or false.
SweepDirection	ST_SweepDirection	required			Specifies the direction in which the arc is drawn. Valid values are Clockwise and Counterclockwise.

	IsStroked	ST_Boolean		true		Specifies whether the stroke for this segment of the path is drawn. Can be true or false.
annotation	Represents an elliptical arc between two points.					

The <ArcSegment> element describes an elliptical arc. It is geometrically defined by the intersection of two ellipses that have the same *x* radius and *y* radius. The ellipses intersect at the starting and ending points of the arc.

Table 11–9. Arc segment definition

Term	**Description**
Starting Point	Implicitly defined by the previous point in the path figure definition.
Ending Point	Specified by the Point attribute.
Arc Size	Defined by the Size attribute. This value consists of the comma-delimited *x* and *y* radii of the ellipses that will be used to define the arc. [*Example*: “100,50” *end example*]
Rotation Angle	Specified by the RotationAngle attribute, this determines how the ellipses defining the arc are rotated with respect to the *x* axis, in degrees. Positive values are clockwise and negative values are counter-clockwise.
Large Arc Flag	Specified by the IsLargeArc attribute, this flag indicates which of the arc pairs created by the intersecting ellipses to use. When the flag is true, it uses the larger arc (arc length >= 180°), and when it is false it uses the smaller arcs (arc length < 180°).
Sweep Direction	Specified by the SweepDirection attribute, this flag determines which of the two possible arcs (selected by the Large Arc Flag) is used. Beginning at the starting point, one arc proceeds in the positive (clockwise) direction, while the other proceeds in the negative (counter-clockwise) direction.

Figure 11–3. Arc choice A

IsLargeArc = false; SweepDirection = Counterclockwise

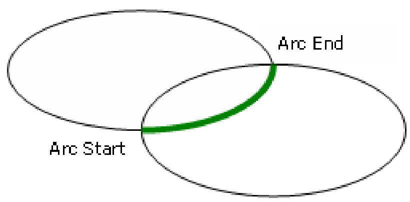


Figure 11–4. Arc choice B

IsLargeArc = false; SweepDirection = Clockwise

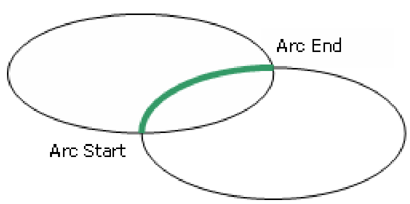


Figure 11–5. Arc choice C

IsLargeArc = true; SweepDirection = Counterclockwise

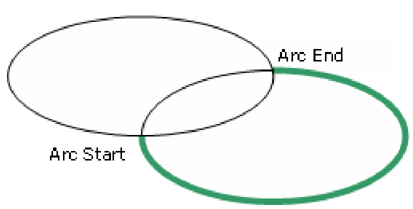
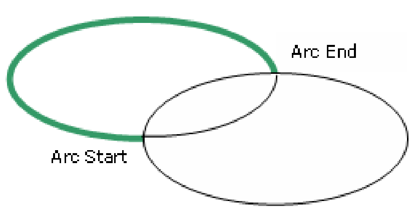


Figure 11–6. Arc choice D

IsLargeArc = true; SweepDirection = Clockwise

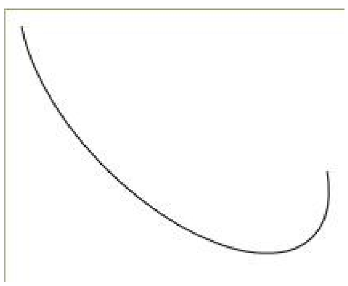


Example 11–5. <ArcSegment> usage

```
<Path Stroke="#000000" StrokeThickness="1">
   <Path.Data>
```

```
        <PathGeometry>
           <PathFigure StartPoint="10,10">
              <ArcSegment
                 Size="100,50"
                 RotationAngle="45"
                 IsLargeArc="true"
                 SweepDirection="Counterclockwise"
                 Point="200,100" />
           </PathFigure>
        </PathGeometry>
     </Path.Data>
  </Path>
```

This markup generates the following arc:



end example]

11.2.2.2.1 Out-of-Range Attributes

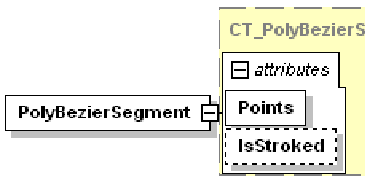
The following guidelines are followed when encountering incompatible attribute values on an <ArcSegment> element:

- If the arc is impossible to render given the combination of radii specified in the **Size** attribute and the angle of rotation specified in the **RotationAngle** attribute, the ellipses are scaled equally until there is exactly one solution that satisfies the arc requirements to pass through the specified **Point** attribute.
- If the **Point** attribute is the same as the previous point in the path figure, the segment is omitted.
- If either the *x* or *y* radius in the **Size** attribute is 0, the segment is rendered as a poly line segment with a single line segment to the *x,y* coordinates specified by the **Point** attribute.
- The *x* or *y* radius in the **Size** attribute MUST NOT be negative [M2.72].
- If the **RotationAngle** value is greater than 360, it is replaced by the value of the **RotationAngle** modulo 360. If it is less than 0, it is replaced with a value normalized to the range 0–360.

11.2.2.3 <PolyBezierSegment> Element

element **PolyBezierSegment**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
Points	ST_Points	required			Specifies control points for multiple Bézier segments. Coordinate values within each pair are comma-separated and additional whitespace may appear. Coordinate pairs are separated from other coordinate pairs by whitespace.
IsStroked	ST_Boolean		true		Specifies whether the stroke for this segment of the path is drawn. Can be true or false.

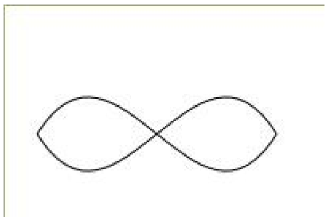
annotation: A series of Bézier segments.

The <PolyBezierSegment> element describes a set of cubic Bézier curves. Bézier curves are drawn from the previous point in the path figure or the previous Bézier curve in the segment and terminate at the third point (x_{3n},y_{3n}) in the **Points** attribute (where *n* is the curve being drawn). The tangents and curvature of each Bézier curve are controlled by the first two control points (x_{3n-2},y_{3n-2} and x_{3n-1},y_{3n-1}) in the **Points** attribute. The **Points** attribute contains a multiple of three whitespace-delimited pairs of comma-delimited *x,y* values.

Example 11–6. <PolyBezierSegment> usage

```
<Path Stroke="#000000" StrokeThickness="1">
   <Path.Data>
      <PathGeometry>
         <PathFigure StartPoint="20,80">
            <PolyBezierSegment Points="70,0 120,160 170,80 120,0 70,160
               20,80" />
         </PathFigure>
      </PathGeometry>
   </Path.Data>
</Path>
```

This markup generates the following results:

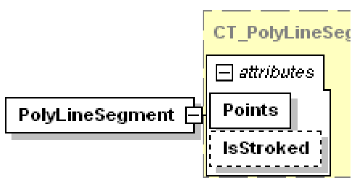


end example]

11.2.2.4 <PolyLineSegment> Element

element **PolyLineSegment**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
Points	ST_Points	required			Specifies a set of coordinates for the multiple segments that define the poly line segment. Coordinate values within each pair are comma-separated and additional whitespace may appear. Coordinate pairs are separated from other coordinate pairs by whitespace.
IsStroked	ST_Boolean		true		Specifies whether the stroke for this segment of the path is drawn. Can be true or false.

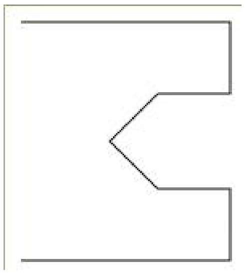
annotation: Specifies a set of points between which lines are drawn.

The <PolyLineSegment> element describes a polygonal drawing containing an arbitrary number of individual vertices. The **Points** attribute defines the vertices and contains whitespace-delimited pairs of comma-delimited *x,y* values.

Example 11–7. <PolyLineSegment> usage

```
<Path Stroke="#000000" StrokeThickness="1">
   <Path.Data>
      <PathGeometry>
         <PathFigure StartPoint="10,10">
            <PolyLineSegment Points="140,10 140,55 95,55 65,85 95,115
               140,115 140,160 10,160" />
         </PathFigure>
      </PathGeometry>
   </Path.Data>
</Path>
```

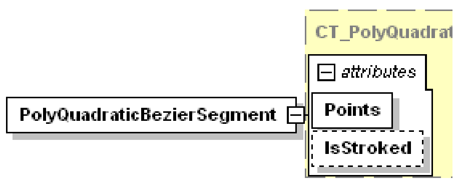
This markup produces the following figure:



end example]

11.2.2.5 <PolyQuadraticBezierSegment> Element

element **PolyQuadraticBezierSegment**

diagram	
attributes	see below
annotation	A series of quadratic Bézier segments.

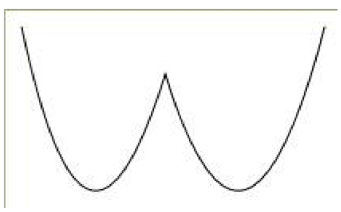
Name	Type	Use	Default	Fixed	Annotation
Points	ST_Points	required			Specifies control points for multiple quadratic Bézier segments. Coordinate values within each pair are comma-separated and additional whitespace may appear. Coordinate pairs are separated from other coordinate pairs by whitespace.
IsStroked	ST_Boolean		true		Specifies whether the stroke for this segment of the path is drawn. Can be true or false.

The <PolyQuadraticBezierSegment> element describes a set of quadratic Bézier curves from the previous point in the path figure through a set of vertices, using specified control points. The **Points** attribute defines an off-curve control point (x_{2n-1},y_{2n-1}) followed by the end point (x_{2n},y_{2n}) for each quadratic Bézier curve (where *n* represents the quadratic Bézier curve). The **Points** attribute contains a multiple of two whitespace-delimited pairs of comma-delimited *x,y* values.

Example 11–8. <PolyQuadraticBezierSegment> usage

```
<Path Stroke="#000000" StrokeThickness="1">
   <Path.Data>
      <PathGeometry>
         <PathFigure StartPoint="10,10">
            <PolyQuadraticBezierSegment Points="50,200 100,40 150,200
               200,10" />
         </PathFigure>
      </PathGeometry>
   </Path.Data>
</Path>
```

This markup produces the following curve:



end example]

11.2.2.6 Closed <PathFigure>

If the **IsClosed** attribute of the <PathFigure> element is set to true, a straight line is drawn from the last point in the last segment of the <PathFigure> element to the **StartPoint** attribute of the <PathFigure> element. If the **IsClosed** attribute is omitted, its default setting is "false".

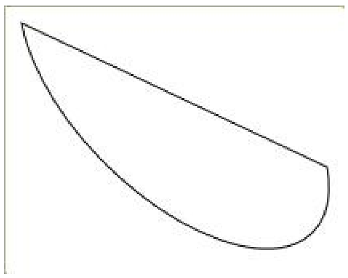
<PathFigure> elements used in filled <Path> elements or as **Clip** attributes are implicitly closed.

Example 11–9. Closed <PathFigure> usage

The following markup shows the arc segment as shown in Example 11–5 with the **IsClosed** attribute of the <PathFigure> element set to true.

```
<Path Stroke="#000000" StrokeThickness="1">
   <Path.Data>
      <PathGeometry>
         <PathFigure StartPoint="10,10" IsClosed="true">
            <ArcSegment
               Size="100,50"
               RotationAngle="45"
               IsLargeArc="true"
               SweepDirection="Counterclockwise"
               Point="200,100" />
         </PathFigure>
      </PathGeometry>
   </Path.Data>
</Path>
```

This markup generates the following figure:



end example]

11.2.3 Abbreviated Geometry Syntax

Abbreviated geometry syntax MAY be used to specify a geometry of one or more figures comprised of multiple segments [M2.72]. A geometry is specified with an optional FillRule command (not allowed in the **Figures** attribute of a <PathGeometry> element) followed by one or more figure definitions. Figure definitions are specified with a Move command, a set of drawing commands to create segments, and an optional Close command to create a closing segment. Drawing commands include:

- Line
- Horizontal Line
- Vertical Line
- Cubic Bézier Curve
- Quadratic Bézier Curve
- Smooth Cubic Bézier Curve
- Elliptical Arc

A command is represented by a single letter and is followed by zero or more whitespace characters, which are followed by command parameters. Parameters are whitespace-delimited. Points are specified as a comma-delimited pair with zero or more whitespace characters.

Uppercase letters denote absolute values and lowercase letters denote relative values. When relative coordinate values are specified, each coordinate pair expresses an offset relative to the current endpoint (the previous command’s terminating coordinate pair). If a relative value is used for the first Move command, the current endpoint is, by definition, 0,0.

If a relative value is used following a Close command, the current endpoint is the first point of the previous figure.

If entering more than one drawing command of the same type sequentially, the duplicate command entry MAY be omitted [M2.72]. [*Example*: “L 100,200 300,400” is equivalent to “L 100,200 L 300,400”. *end example*] The current endpoint is determined as though each command appeared individually.

Values specifying coordinates can be real numbers.

For more information, see F, “Abbreviated Geometry Syntax Algorithm.”

Table 11–10. Commands

Name	**Syntax**	**Description**	**Non-Abbreviated Equivalent**
FillRule	`F fFillRule`	Establishes the fill rule that should be used for this geometry. A value of 0 is equivalent to a FillRule value of EvenOdd; a value of 1 is equivalent to a FillRule value of NonZero. The default value if this command is omitted is 0. This command MUST appear only as the first command in the abbreviated geometry syntax [M2.72]. This command MUST NOT be specified in the value of the Figures attribute of the <PathGeometry> element [M2.72]. [*Example*: `F 0` *end example*]	<PathGeometry> FillRule attribute

Move	`M x,y`
	or
	`m x,y`
		Establishes a new current endpoint. Every geometry MAY specify one or more figures, and MAY be preceded by a FillRule command where allowed [M2.72]. The first figure in a geometry MUST begin with a Move command [M2.72]. Subsequent Move commands indicate the start of a new figure but MAY be omitted, indicating the current endpoint for the subsequent figure is the same as the end point of the previous figure [M2.72]. [*Example*: M 1.0,1.5 *end example*]	<PathFigure> StartPoint attribute

Line	`L x,y`
	or
	`l x,y`
		Draws a straight line from the current point to the specified point. [*Example*: `L 20,30` *end example*]	<PolyLineSegment> element

Horizontal Line	`H x`
	or
	`h x`
		Draws a horizontal line from the current endpoint to the specified *x* coordinate. [*Example*: `H 90` *end example*]	<PolyLineSegment> element

Name	**Syntax**	**Description**	**Non-Abbreviated Equivalent**

Vertical Line	`V y`
	or
	`v y`
		Draws a vertical line from the current endpoint to the specified *y* coordinate. [*Example*: `v 90` *end example*]	<PolyLineSegment> element

Cubic Bézier Curve	C x_1,y_1 x_2,y_2 x_3,y_3 or c x_1,y_1 x_2,y_2 x_3,y_3	Draws a cubic Bézier curve from the current endpoint to the specified point (x_3,y_3) using the two specified control points (x_1,y_1 and x_2,y_2). The first control point determines the initial direction (tangent) of the curve, and the second determines the terminating direction (tangent) of the curve. [*Example*: `C 100,200 200,400 300,200` *end example*]	<PolyBezierSegment> element
Quadratic Bézier Curve	Q x_1,y_1 x_2,y_2 or q x_1,y_1 x_2,y_2	Draws a quadratic Bézier curve from the current endpoint to the specified point (x_2,y_2) using the specified control point (x_1,y_1). [*Example*: `q 100,200 300,200` *end example*]	<PolyQuadraticBezierSegment> element
Smooth Cubic Bézier Curve	S x_1,y_1 x_2,y_2 or s x_1,y_1 x_2,y_2	Draws a cubic Bézier curve from the current endpoint to the specified point (x_2,y_2). The first control point is assumed to be the reflection of the second control point of the previous command, relative to the current endpoint. If there is no previous command or if the previous command was not a Cubic Bézier Curve command or Smooth Cubic Bézier Curve command, the first control point is assumed to be coincident with the current endpoint. The second control point is specified by x_1,y_1. [*Example*: `S 100,200 200,300` *end example*]	<PolyBezierSegment> element
Elliptical Arc	A x_r,y_r r_x fArc fSweep x,y or a x_r,y_r r_x fArc fSweep x,y	Draws an elliptical arc from the current endpoint to the specified point (x,y). The size and orientation of the ellipse are defined by x_r,y_r. r_x,x_r defines the *x* radius, y_r defines	<ArcSegment> element

Name	Syntax	Description	Non-Abbreviated Equivalent
		the *y* radius, and r_x defines the *x*-axis rotation in degrees, which indicates how the ellipse is rotated relative to the current coordinate system. The center of the ellipse is calculated automatically. In most situations, four different arcs satisfy the specified constraints. `fArc` and `fSweep` indicate which arc to use. Of the four candidate arc sweeps, two represent large arcs with sweeps of 180° or greater, and two represent smaller arcs with sweeps less than 180°. If `fArc` is 1, one of the two larger arc sweeps is chosen. If `fArc` is 0, one of the smaller arc sweeps is chosen. No other values of `fArc` are valid. If `fSweep` is 1, the arc is drawn in a positive-angle (clockwise) direction. If `fSweep` is 0, the arc is drawn in a negative-angle (counter-clockwise) direction. No other values of `fSweep` are valid. [*Example*: `a 200,70 10 0 1 100,100` *end example*]	
Close	`Z` or `z`	Draws a straight line from the current endpoint to the first point of the current figure and then ends the figure. If the command following a Close command is a Move command, the Move command specifies the initial point of the next figure. Otherwise, the next figure starts at the same initial point as the current figure.	<PathFigure> IsClosed attribute

Example 11–10. A path described using abbreviated syntax

The following markup demonstrates a simple path, which is drawn using the abbreviated syntax:

```
<Path Stroke="#000000" Data="M 100,100 L 300,100 L 200,300 z" />
```

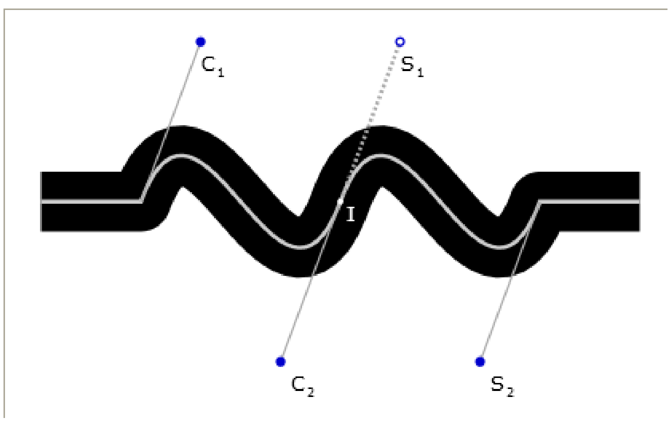
end example]

11.2.3.1 Smooth Bézier Curve Abbreviated Syntax

Smooth Bézier curves specified with the abbreviated geometry syntax are basic cubic Bézier curves with an implied first control point. This control point is coincident with the endpoint of the previous segment unless the previous segment is also a Bézier curve. In this case, the first control point of the smooth Bézier curve is a reflection of the second control point of the previous curve segment around the start point of the smooth Bezier curve segment, as shown below.

Example 11–11. Smooth Bézier curve

In the following example, C_1 and C_2 represent the first and second control points of the first cubic Bézier curve segment, respectively. S_1 represents the implied first control point of the smooth Bézier curve segment. S_2 represents the specified control point of the smooth Bézier curve segment. I represents the inflection point around which control point S_1 is derived from control point C_2.



The above diagram is generated with the following markup:

```
<Canvas RenderTransform="1.25,0,0,1.25,-40,20" >
   <!-- Main Path -->
   <Path Stroke="#000000" StrokeThickness="30" StrokeLineJoin="Round"
   Data="M50,80 L100,80 C130,0 170,160 200,80 S270,160 300,80 L350,80"/>
   <Path Stroke="#CCCCCC" StrokeThickness="2"
   Data="M50,80 L100,80 C130,0 170,160 200,80 S270,160 300,80 L350,80"/>
   <!-- C1 -->
   <Path Stroke="#AAAAAA" StrokeThickness="1" Data="M 100,80 L 130,0" />
   <Path Stroke="#0000CC" StrokeThickness="5" StrokeStartLineCap="Round"
      StrokeEndLineCap="Round" Data="M 130,0 L 130,0" />
   <Glyphs Fill="#000000" UnicodeString="C" OriginX="130" OriginY="15"
      FontUri="../Resources/Fonts/Verdana.ttf" FontRenderingEmSize="10"
   />
   <Glyphs Fill="#000000" UnicodeString="1" OriginX="138" OriginY="18"
```

```
         FontUri="../Resources/Fonts/Verdana.ttf" FontRenderingEmSize="6"
      />
      <!-- C2 -->
      <Path Stroke="#AAAAAA" Data="M 200,80 L 170,160" />
      <Path Stroke="#0000CC" StrokeThickness="5" StrokeStartLineCap="Round"
         StrokeEndLineCap="Round" Data="M 170,160 L 170,160" />
      <Glyphs Fill="#000000" UnicodeString="C" OriginX="175"
         OriginY="175" FontUri="../Resources/Fonts/Verdana.ttf"
         FontRenderingEmSize="10" />
      <Glyphs Fill="#000000" UnicodeString="2" OriginX="183"
         OriginY="178" FontUri="../Resources/Fonts/Verdana.ttf"
         FontRenderingEmSize="6" />
      <!-- S1 -->
      <Path Stroke="#AAAAAA" StrokeThickness="2"
         StrokeDashArray="0.75 0.75" Data="M 200,80 L 230,0" />
      <Path Stroke="#0000CC" StrokeThickness="5" StrokeStartLineCap="Round"
         StrokeEndLineCap="Round" Data="M 230,0 L 230,0" />
      <Path Stroke="#FFFFFF" StrokeThickness="3" StrokeStartLineCap="Round"
         StrokeEndLineCap="Round" Data="M 230,0 L 230,0" />
      <Glyphs Fill="#000000" UnicodeString="S" OriginX="230" OriginY="15"
         FontUri="../Resources/Fonts/Verdana.ttf" FontRenderingEmSize="10"
      />
      <Glyphs Fill="#000000" UnicodeString="1" OriginX="238" OriginY="18"
         FontUri="../Resources/Fonts/Verdana.ttf" FontRenderingEmSize="6"
      />
      <!-- S2 -->
      <Path Stroke="#AAAAAA" StrokeThickness="1" Data="M 300,80 L 270,160"
      />
      <Path Stroke="#0000CC" StrokeThickness="5" StrokeStartLineCap="Round"
         StrokeEndLineCap="Round" Data="M 270,160 L 270,160" />
      <Glyphs Fill="#000000" UnicodeString="S" OriginX="275" OriginY="175"
         FontUri="../Resources/Fonts/Verdana.ttf" FontRenderingEmSize="10"
      />
      <Glyphs Fill="#000000" UnicodeString="2" OriginX="283" OriginY="178"
         FontUri="../Resources/Fonts/Verdana.ttf" FontRenderingEmSize="6"
      />
      <!-- Inflection -->
      <Path Stroke="#FFFFFF" StrokeThickness="3" StrokeStartLineCap="Round"
         StrokeEndLineCap="Round" Data="M 200,80 L 200,80" />
      <Glyphs Fill="#FFFFFF" UnicodeString="I" OriginX="203" OriginY="90"
         FontUri="../Resources/Fonts/Verdana.ttf" FontRenderingEmSize="10"
      />
</Canvas>
```

end example]

11.2.3.2 Relative Commands and Curve Control Points

When using relative (lowercase) commands with the abbreviated geometry syntax, each control point and end point are individually specified relative to the start point of that segment.

Example 11–12. Relative commands and curves

The following markup describes a simple shape using cubic Bézier curves:

```
<Path Stroke="#000000" Data="M 50,20 L 150,20 C 250,75 170,130 120,100
   C 70,70 90,110 130,160 Q 0,150 50,20" />
```

This markup describes the same shape, using relative commands:

```
<Path Stroke="#000000" Data="M 50,20 l 100,0 c 100,55 20,110 -30,80
   c -50,-30 -30,10 10,60 q -130,-10 -80,-140" />
```

end example]

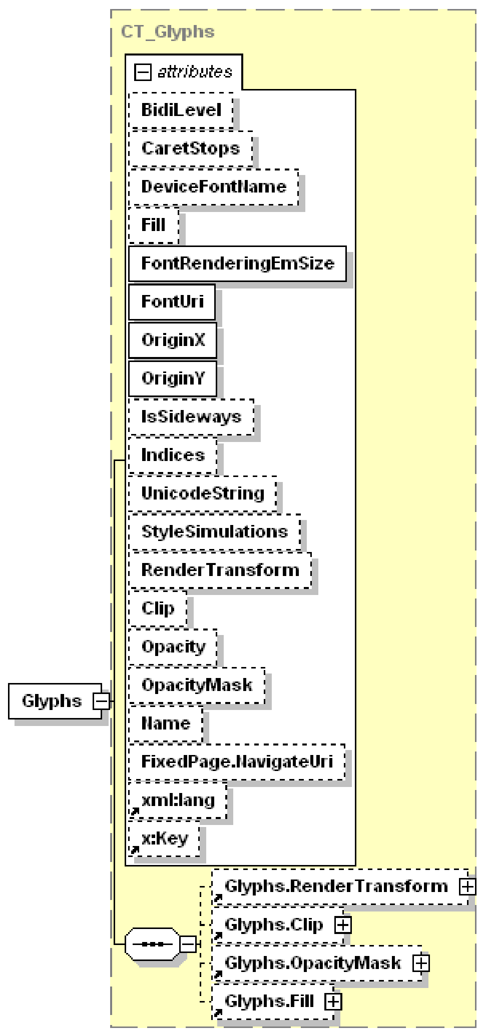
12. Text

A run of text sharing the same characteristics is represented by a <Glyphs> element. Text runs are broken by line advances and formatting changes. The set of properties on the <Glyphs> element allows for a complete description of the glyph characteristics, such as the fill and opacity, as well as clipping information. The <Glyphs> element allows specification of a Unicode string and supports bidirectional and vertical text.

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 may 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, "Common Properties."

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

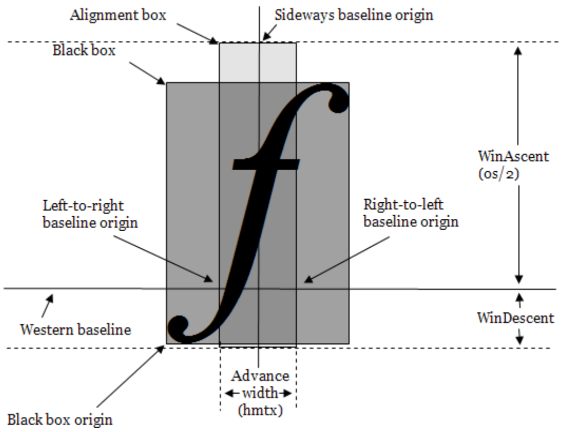


Figure 12–2. Upright (usually horizontal) glyph metrics

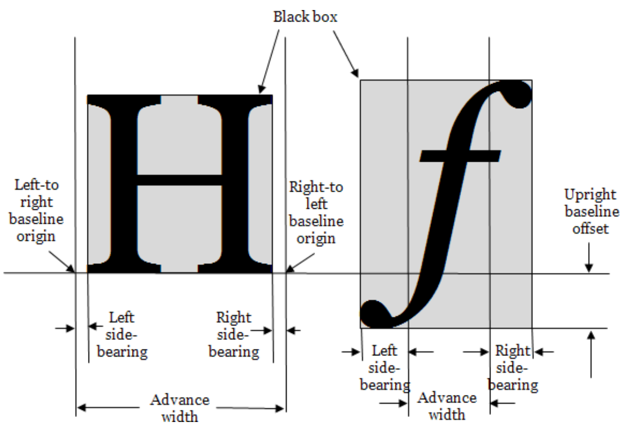
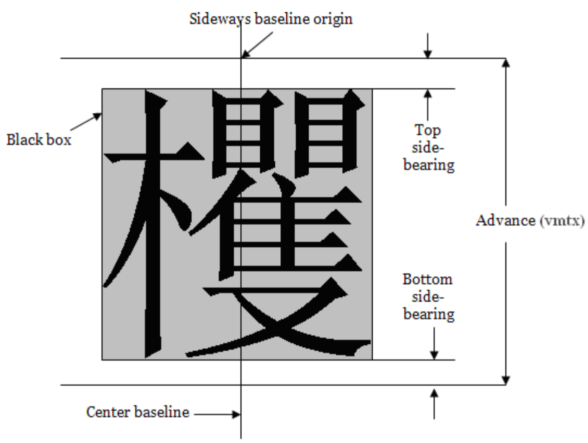


Figure 12–3. Sideways (usually vertical) glyph metrics



In general, glyphs within a font are either base glyphs or combining marks that may 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, may 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 may map to two UTF-16 code units, multiple UTF-16 code units may map to a single glyph, single UTF-16 code units may map to multiple glyphs based on context, and multiple UTF-16 code units may map indivisibly to multiple glyphs. In these cases, the clusters of UTF-16 code units are mapped using a cluster map.

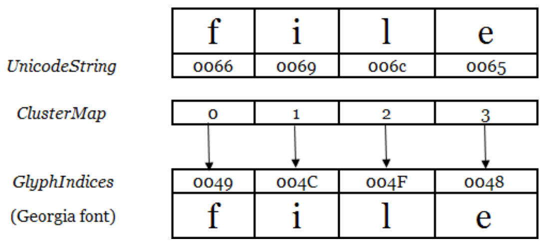
The cluster map contains one entry for each UTF-16 code unit in the **UnicodeString** attribute. Each entry specifies the offset of the first glyph that represents the cluster of UTF-16 code units.

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.



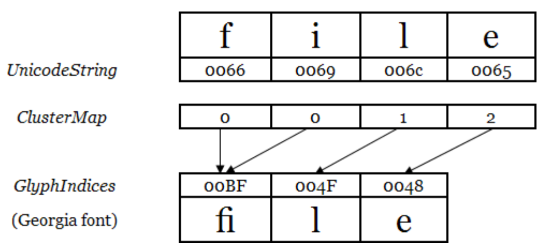
end example]

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.



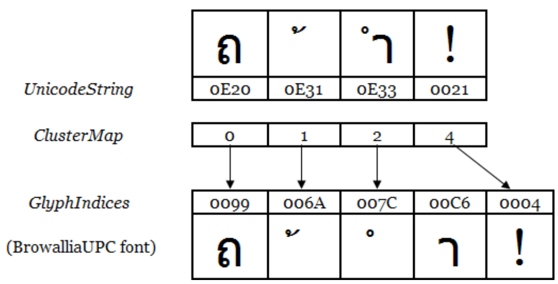
end example]

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

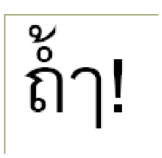
The Thai *Sara Am* character contains a part that sits on top of the previous base character (the ring), and a part that sits to the right of the base character (the hook). When Thai text is micro-justified, the hook is spaced apart from the base character, while the ring remains on top of the base character.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



The markup appears as follows:

```
<Glyphs
   FontUri="../Resources/Fonts/browau.ttf"
   UnicodeString="&#xe20;&#xe49;&#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:



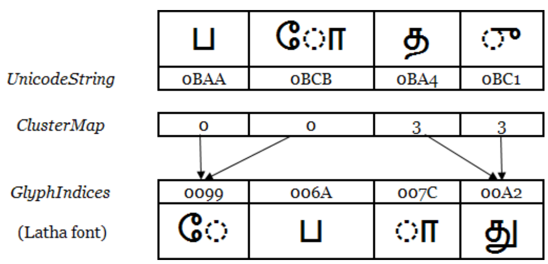
end example]

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. If the **Indices** attribute specifies a GlyphIndex that does not exist in the font, the consumer MUST generate an error [M5.4]. 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 12–1. 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 12–1. 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 12–2. 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 may be specified. Default value is 1.
ClusterGlyphCount	Number of glyph indices that combine to form this cluster.

	One or more indices may be specified. Default value is 1.

Example 12–2. 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, "Non-Standard Font Compatibility Encoding," on page 35 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 **IsSideway** 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 may 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].

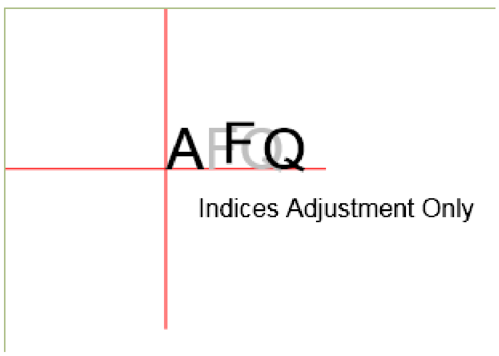
Table 12–3. IsSideways and BidiLevel effects on origin placement

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 12–3. 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" />
```

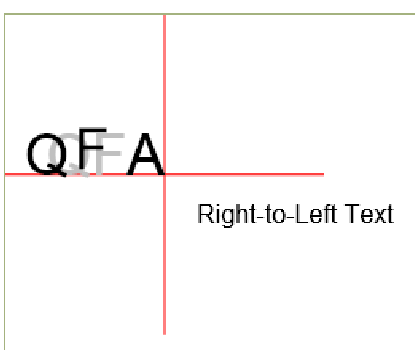


end example]

Example 12–4. 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" />
```

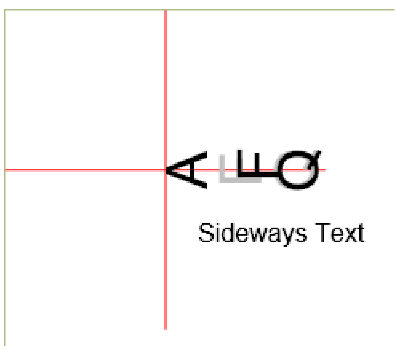


end example]

Example 12–5. Sideways text (IsSideways set to true)

This example shows the **IsSideways** attribute set to true. The BidiLevel must be even when the **IsSidways** 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" />
```

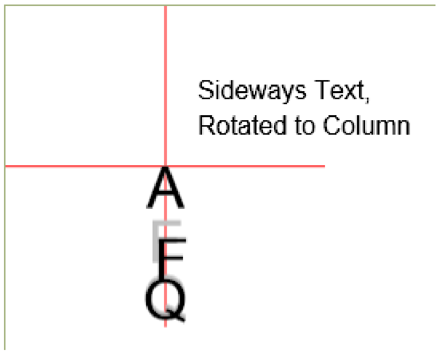


end example]

Example 12–6. 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, "Positioning Content," on page 185.

```
<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 12–7. 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, "Positioning Content," on page 185) 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:

これは、縦書きの日本語テキストが
どのように列で書かれるかの例です。
テキストは縦に読み、一行ずつ進みます。
他の言語も縦書きで書かれます。

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 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 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 may 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, "Language," on page 47.

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 12–8. 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 section 12.1.6.1, “Calculating Sideways Text Origin and Advance Width” (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 section 12.1.6.1, “Calculating Sideways Text Origin and Advance Width” (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 12–9. 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 12–10. 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:

Simulated italic font...

end example]

Example 12–11. 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:

WAVE (no kerning)

WAVE (with kerning)

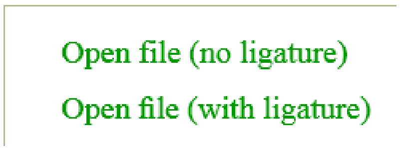
end example]

Example 12–12. 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 12–13. 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:

ёжик в тумане

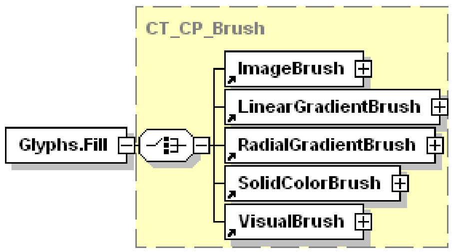
ёжик в тумане

енамут в кижё

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 may be used.

13. Brushes

Brushes are used to paint the interior of the geometric shapes defined by a <Path> element and the characters rendered with a <Glyphs> element. They are also used to define the alpha-transparency mask in the <Canvas.OpacityMask>, <Path.OpacityMask>, and <Glyphs.OpacityMask> property elements.

All brushes are defined relative to a coordinate space. Most brushes (including image brushes, visual brushes, linear gradient brushes, and radial gradient brushes) may specify a coordinate-space transform, in which the Transform property is concatenated with the current effective coordinate space to yield an effective coordinate space local to the brush. For image brushes and visual brushes, the viewport is transformed using the local effective render transform. For linear gradient brushes, the start point and end point are transformed. For radial gradient brushes, the ellipse defined by the center, *x* radius, *y* radius, and gradient origin is transformed.

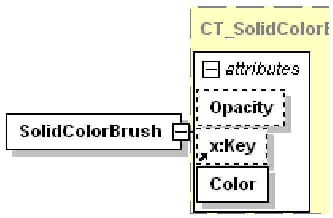
Table 13–1. Brush types

Name	Description
Solid color brush	Fills a region with a solid color
Image brush	Fills a region with an image
Visual brush	Fills a region with a drawing
Linear gradient brush	Fills a region with a linear gradient
Radial gradient brush	Fills a region with a radial gradient

13.1 <SolidColorBrush> Element

element **SolidColorBrush**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
Opacity	ST_ZeroOne		1.0		Defines the uniform transparency of the brush fill. Values range from 0 (fully transparent) to 1 (fully opaque), inclusive. Values outside of this range are invalid.
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 [M6.1].
Color	ST_Color	required			Specifies the color for filled elements. An sRGB color value specified as a 6-digit hexadecimal number (#RRGGBB) or an extended color.

annotation: Fills defined geometric regions with a solid color.

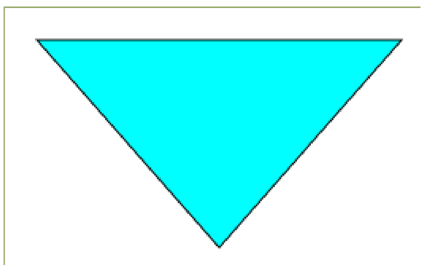
The <SolidColorBrush> element is used to fill defined geometric regions with a solid color. If there is an alpha component of the color, it is combined in a multiplicative way with the corresponding **Opacity** attribute.

Example 13–1. <SolidColorBrush> usage

The following markup illustrates how a solid color brush fills a path.

```
<Path Stroke="#000000">
   <Path.Fill>
      <SolidColorBrush Color="#00FFFF" />
   </Path.Fill>
   <Path.Data>
      <PathGeometry>
         <PathFigure StartPoint="20,20" IsClosed="true">
            <PolyLineSegment Points="250,20 135,150" />
         </PathFigure>
      </PathGeometry>
   </Path.Data>
</Path>
```

This markup is rendered as follows:

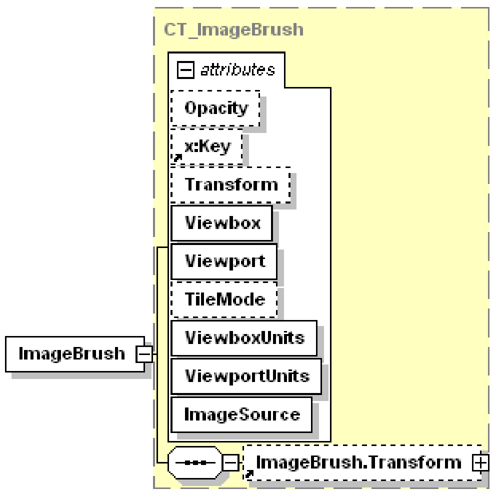


end example]

13.2 <ImageBrush> Element

element **ImageBrush**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
Opacity	ST_ZeroOne		1.0		Defines the uniform transparency of the brush fill. Values range from 0 (fully transparent) to 1 (fully opaque), inclusive. Values outside of this range are invalid.
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 [M6.2].
Transform	ST_RscRefMatrix				Describes the matrix transformation applied to the coordinate space of the brush. The Transform property is concatenated with the current effective render transform to yield an effective render transform local to the brush. The viewport for the brush is transformed using the local effective render transform.
Viewbox	ST_ViewBox	required			Specifies the position and dimensions of the brush's source content. Specifies four comma-

						separated real numbers (x, y, Width, Height), where width and height are non-negative. The dimensions specified are relative to the image's physical dimensions expressed in units of 1/96". The corners of the viewbox are mapped to the corners of the viewport, thereby providing the default clipping and transform for the brush's source content.
	Viewport	ST_ViewBox	required			Specifies the region in the containing coordinate space of the prime brush tile that is (possibly repeatedly) applied to fill the region to which the brush is applied. Specifies four comma-separated real numbers (x, y, Width, Height), where width and height are non-negative. The alignment of the brush pattern is controlled by adjusting the x and y values.
	TileMode	ST_TileMode		None		Specifies how contents will be tiled in the filled region. Valid values are None, Tile, FlipX, FlipY, and FlipXY.
	ViewboxUnits	ST_ViewUnits	required		Absolute	Specifies the relationship of the viewbox coordinates to the containing coordinate space.
	ViewportUnits	ST_ViewUnits	required		Absolute	Specifies the relationship of the viewport coordinates to the containing coordinate space.
	ImageSource	ST_UriCtxBmp	required			Specifies the URI of an image resource or a combination of the URI of an image resource a color profile resource. See the Color clause for important details. The URI MUST refer to parts in the package [M2.1].
annotation	Fills a region with an image.					

The <ImageBrush> element is used to fill a region with an image. The image is defined in a coordinate space specified by the resolution of the image. The image MUST refer to a JPEG, PNG, TIFF 6.0, or Windows Media Photo image part within the

XPS Document package [M6.3]. For more information, see §9.1.5, “Image Parts,” on page 24. A URI part name for the image is specified using the **ImageSource** attribute.

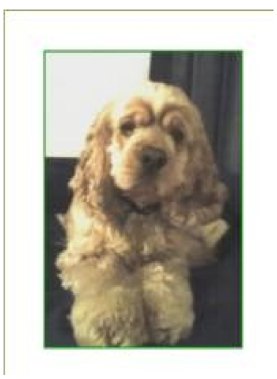
Image brushes share a number of tile-related properties with visual brushes. For details, see §13.4, “Common Attributes for Tiling Brushes,” on page 136.

Example 13–2. <ImageBrush> usage

The following markup describes an image on a canvas.

```
<Canvas>
   <Path Stroke="#008000">
      <Path.Fill>
         <ImageBrush
            ImageSource="dog.jpg"
            TileMode="None"
            Viewbox="0,0,270,423"
            ViewboxUnits="Absolute"
            Viewport="25,25,125,185"
            ViewportUnits="Absolute" />
      </Path.Fill>
      <Path.Data>
         <PathGeometry>
            <PathFigure StartPoint="25,25" IsClosed="true">
               <PolyLineSegment Points="150,25 150,210 25,210" />
            </PathFigure>
         </PathGeometry>
      </Path.Data>
   </Path>
</Canvas>
```

This markup produces the following results:

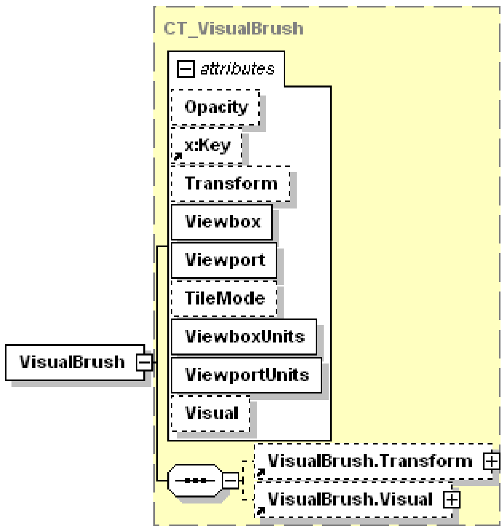


end example]

13.3 <VisualBrush> Element

element **VisualBrush**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
Opacity	ST_ZeroOne		1.0		Defines the uniform transparency of the brush fill. Values range from 0 (fully transparent) to 1 (fully opaque), inclusive. Values outside of this range are invalid.
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 [M6.4].
Transform	ST_RscRefMatrix				Describes the matrix transformation applied to the coordinate space of the brush. The Transform property is concatenated with the current effective render transform to yield an effective render transform local to the brush. The viewport for the brush is

						transformed using that local effective render transform.
	Viewbox	ST_ViewBox	required			Specifies the position and dimensions of the brush's source content. Specifies four comma-separated real numbers (x, y, Width, Height), where width and height are non-negative. The viewbox defines the default coordinate system for the element specified in the <VisualBrush.Visual> property element. The corners of the viewbox are mapped to the corners of the viewport, thereby providing the default clipping and transform for the brush's source content.
	Viewport	ST_ViewBox	required			Specifies the region in the containing coordinate space of the prime brush tile that is (possibly repeatedly) applied to fill the region to which the brush is applied. Specifies four comma-separated real numbers (x, y, Width, Height), where width and height are non-negative. The alignment of the brush pattern is controlled by adjusting the x and y values.
	TileMode	ST_TileMode		None		Specifies how contents will be tiled in the filled region. Valid values are None, Tile, FlipX, FlipY, and FlipXY.
	ViewboxUnits	ST_ViewUnits	required		Absolute	Specifies the relationship of the viewbox coordinates to the containing coordinate space.
	ViewportUnits	ST_ViewUnits	required		Absolute	Specifies the relationship of the viewport coordinates to the containing coordinate space.
	Visual	ST_RscRef				Specifies resource reference to a <Path>, <Glyphs>, or <Canvas> element defined in a resource dictionary and used to draw the brush's source content.

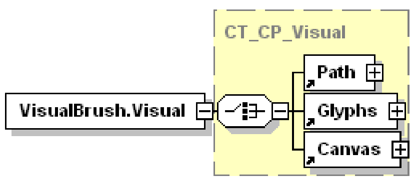
annotation	Fills a region with a drawing. The drawing may be specified as either a child of the <VisualBrush> element, or as a resource reference. Drawing content is expressed using <Canvas>, <Path>, and <Glyphs> elements.

The <VisualBrush> element is used to fill a region with a drawing. The drawing may be specified as either a <VisualBrush.Visual> property element or as a resource reference. Drawing content may include exactly one <Canvas>, <Path>, or <Glyphs> element and that element's child and descendant elements.

Visual brushes share a number of tile-related properties with image brushes. For details, see §13.4, "Common Attributes for Tiling Brushes," on page 136.

13.3.1 <VisualBrush.Visual> Element

element **VisualBrush.Visual**

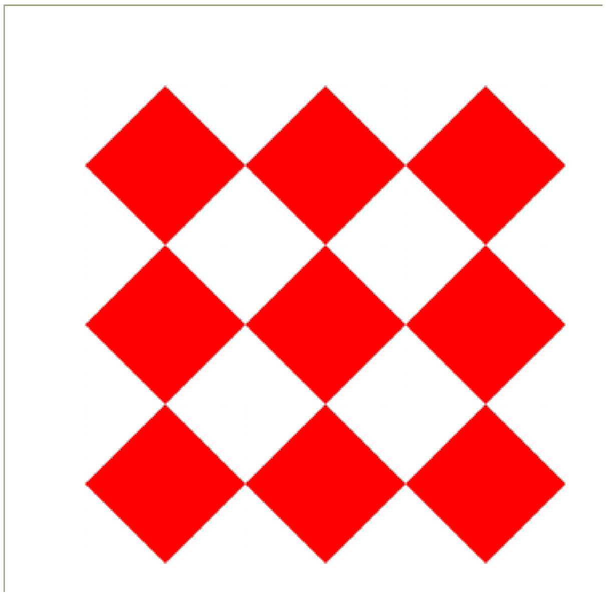
diagram	
annotation	Specifies a <Path> element, <Glyphs> element, or <Canvas> element used to draw the brush's source contents.

The <VisualBrush.Visual> property element contains markup that defines the contents of a single visual brush tile. The tile can be used to fill the geometric region to which the visual brush is applied. The <VisualBrush.Visual> property element contains a single child element. For simple tiles, this can be a single <Path> or <Glyphs> element. More complex visuals containing multiple <Path> and <Glyphs> elements can be grouped within a <Canvas> child element.

Example 13–3. <VisualBrush.Visual> usage

```
<Path>
   <Path.Fill>
      <VisualBrush
         Viewbox="0,0,1,1"
         Viewport="50,50,100,100"
         ViewportUnits="Absolute"
         ViewboxUnits="Absolute"
         TileMode="Tile">
         <VisualBrush.Visual>
            <Path>
               <Path.Fill>
                  <SolidColorBrush Color="#FF0000" />
               </Path.Fill>
               <Path.Data>
                  <PathGeometry>
                     <PathFigure StartPoint="0,0.5" IsClosed="true">
                        <PolyLineSegment Points="0.5,0 1.0,0.5
                           0.5,1.0" />
                     </PathFigure>
                  </PathGeometry>
               </Path.Data>
            </Path>
         </VisualBrush.Visual>
      </VisualBrush>
   </Path.Fill>
   <Path.Data>
      <PathGeometry>
         <PathFigure StartPoint="50,50" IsClosed="true">
            <PolyLineSegment Points="350,50 350,350 50,350" />
         </PathFigure>
      </PathGeometry>
   </Path.Data>
</Path>
```

This markup produces the following result:



end example]

13.4 Common Attributes for Tiling Brushes

Image brushes and Visual brushes share certain tiling characteristics in common. These characteristics are controlled by a common set of attributes described in the table below.

Table 13–2. Common attributes for <ImageBrush> and <VisualBrush> elements

Name	Description
Viewbox	Specifies the region of the source content of the brush that is to be mapped to the viewport.
Viewport	Specifies the position and dimensions of the first brush tile. Subsequent tiles are positioned relative to this tile, as specified by the tile mode.
ViewboxUnits	Specifies the unit type for the Viewbox attribute. MUST have the value “Absolute” [M2.72].
ViewportUnits	Specifies the unit type for the Viewport attribute. MUST have the value “Absolute” [M2.72].
TileMode	Specifies how tiling is performed in the filled geometry. The value is optional, and defaults to “None” if no value is specified.

Both image brushes and visual brushes assume that the background of the brush itself is initially transparent.

13.4.1 Viewbox, Viewport, ViewboxUnits, and ViewportUnits Attributes

The **Viewbox** attribute specifies the portion of a source image or visual to be rendered to the page as a tile. The **Viewport** attribute specifies the dimensions and location, in the effective coordinate space, of the initial tile that will be filled with the specified image or visual fragment. In other words, the **Viewport** attribute defines the initial tile whose origin (x and y values of the top left corner of the tile relative to the current effective render transform) is specified by the first two parameters and whose size (width and height values) is specified by the last two parameters. The tile is then used to fill the geometry specified by the parent element according to the **TileMode** attribute relative to the initial tile.

For images, the dimensions specified by the viewbox are expressed in units of 1/96". The pixel coordinates in the source image are calculated as follows:

```
SourceLeft = HorizontalImageResolution * Viewbox.Left / 96
SourceTop = VerticalImageResolution * Viewbox.Top / 96
SourceWidth = HorizontalImageResolution * Viewbox.Width / 96
SourceHeight = VerticalImageResolution * Viewbox.Height / 96
```

The image resolution used is that specified in the header or tag information of the image. If no resolution is specified, a default resolution of 96 dpi is assumed. The coordinates of the upper-left corner of the image are 0,0.

The viewbox can specify a region larger than the image itself, including negative values.

Example 13–4. ViewboxUnits and ViewportUnits attribute usage

The following markup contains an image brush:

```
<ImageBrush
   ImageSource="../Resources/Images/tiger.jpg"
   Viewbox="24,24,48,48"
   ViewboxUnits="Absolute"
   Viewport="96,96,192,192"
   ViewportUnits="Absolute"
   TileMode="None" />
```

Assuming the default fixed page coordinate system and that tiger.jpg specifies a resolution of 50 dpi and measures 100 pixels horizontally and 50 pixels vertically, the physical dimensions of the image are expressed (in units of 1/96") as 96 * 100 / 50 = 192 horizontal and 96 * 50 / 50 = 96 vertical.

The viewbox uses a square starting at 24,24 (a quarter-inch from left and a quarter-inch from top) in the image, and extending for 48,48 (a half-inch to the right and a half-inch down) and scales it to a square starting at one inch from the left edge of the physical page and one inch from the top of the physical page and extending two inches to the right and two inches down. *end example*]

13.4.1.1 Viewbox and Viewport Examples

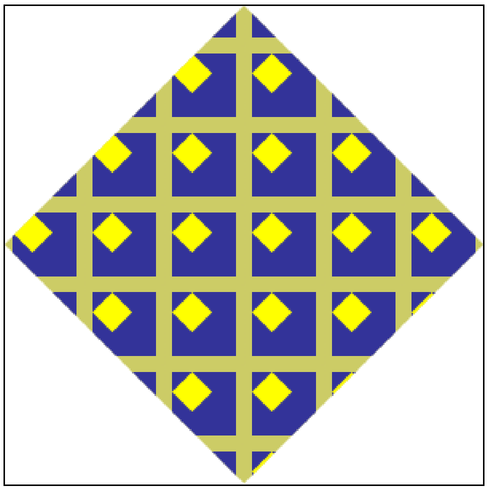
The following examples demonstrate how adjusting the viewbox and viewport can affect output.

Example 13–5. Tiling brush base image and rendering

The following markup describes a base image.

```
<!-- Draw background diamond to show where fill affects background -->
<Path Fill="#CCCC66" Data="M 150,0 L 300,150 L 150,300 L 0,150 Z" />
<Path Data="M 150,0 L 300,150 L 150,300 L 0,150 Z">
   <Path.Fill>
      <VisualBrush
         Viewbox="0,0,1,1"
         Viewport="150,75,50,50"
         ViewboxUnits="Absolute"
         ViewportUnits="Absolute"
         TileMode="Tile">
         <VisualBrush.Visual>
            <Canvas>
               <Path Fill="#333399" Data="M 0.1,0.1 L 0.9,0.1 L 0.9,0.9
                  L 0.1,0.9 Z" />
               <Path Fill="#FFFF00" Data="M 0.1,0.35 L 0.35,0.1
                  L 0.6,0.35 L 0.35,0.6 Z" />
            </Canvas>
         </VisualBrush.Visual>
      </VisualBrush>
   </Path.Fill>
</Path>
```

This markup is rendered as follows:



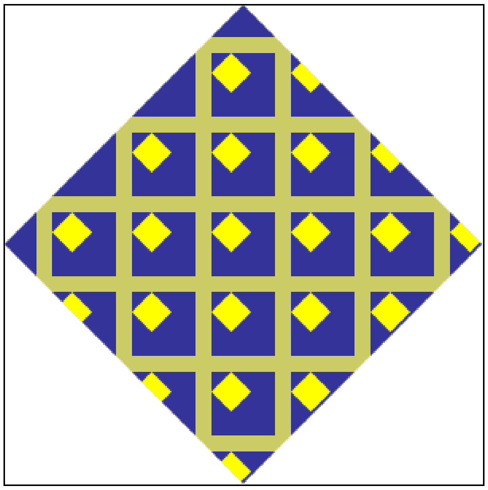
end example]

Example 13–6. Tiling brush Viewport adjustments

By adjusting the viewport, the position of the tiles within the image can be changed:

```
<!-- Draw background diamond to show where fill affects background -->
<Path Fill="#CCCC66" Data="M 150,0 L 300,150 L 150,300 L 0,150 Z" />
<Path Data="M 150,0 L 300,150 L 150,300 L 0,150 Z">
   <Path.Fill>
      <VisualBrush
         Viewbox="0,0,1,1"
         Viewport="125,125,50,50"
         ViewboxUnits="Absolute"
         ViewportUnits="Absolute"
         TileMode="Tile">
         <VisualBrush.Visual>
            <Canvas>
               <Path Fill="#333399" Data="M 0.1,0.1 L 0.9,0.1 L 0.9,0.9
                  L 0.1,0.9 Z" />
               <Path Fill="#FFFF00" Data="M 0.1,0.35 L 0.35,0.1
                  L 0.6,0.35 L 0.35,0.6 Z" />
            </Canvas>
         </VisualBrush.Visual>
      </VisualBrush>
   </Path.Fill>
</Path>
```

This markup is rendered as follows:



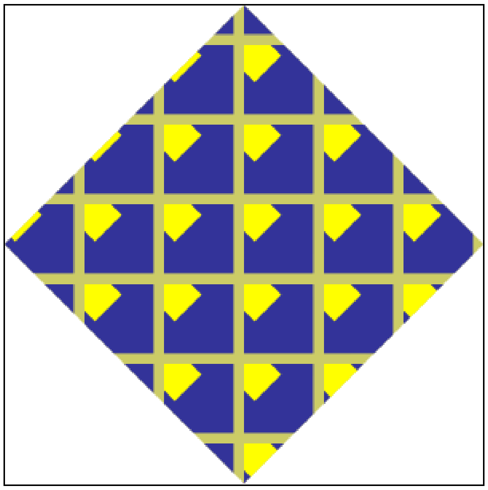
end example]

Example 13–7. Tiling brush viewbox adjustments

The following markup uses a smaller window on the viewbox to zoom in on each tile:

```
<!-- Draw background diamond to show where fill affects background -->
<Path Fill="#CCCC66" Data="M 150,0 L 300,150 L 150,300 L 0,150 Z" />
<Path Data="M 150,0 L 300,150 L 150,300 L 0,150 Z">
   <Path.Fill>
      <VisualBrush
         Viewbox="0.25,0.25,0.75,0.75"
         Viewport="150,75,50,50"
         ViewboxUnits="Absolute"
         ViewportUnits="Absolute"
         TileMode="Tile">
         <VisualBrush.Visual>
            <Canvas>
               <Path Fill="#333399" Data="M 0.1,0.1 L 0.9,0.1 L 0.9,0.9
                  L 0.1,0.9 Z" />
               <Path Fill="#FFFF00" Data="M 0.1,0.35 L 0.35,0.1
                  L 0.6,0.35 L 0.35,0.6 Z" />
            </Canvas>
         </VisualBrush.Visual>
      </VisualBrush>
   </Path.Fill>
</Path>
```

This markup is rendered as follows:



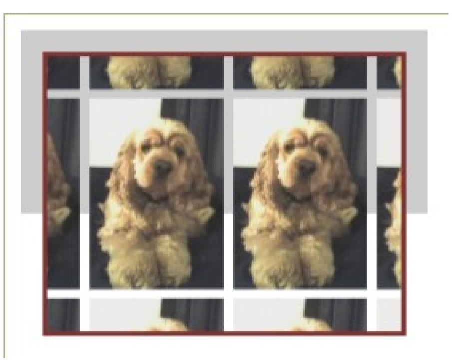
end example]

Example 13–8. Image brush with a Viewbox larger than the image

An image brush can specify a tile with the **Viewbox** attribute that exceeds the size of the image it uses, including negative values, as shown below.

```
<Path Fill="#CCCCCC" Data="M 10,10 L 265,10 L 265,125 L 10,125 Z" />
<Path Stroke="#803333" StrokeThickness="3"
   Data="M 25,25 L 250,25 L 250,200 L 25,200 Z">
   <Path.Fill>
         <ImageBrush ImageSource="../Resources/Images/dog.jpg"
               TileMode="Tile"
               Viewbox="-10,-10,290,443" ViewboxUnits="Absolute"
               Viewport="50,50,90,125" ViewportUnits="Absolute" />
   </Path.Fill>
</Path>
```

This markup is rendered as follows. Note that the area around the image is transparent, revealing the underlying path between the tiles.



end example]

13.4.2 TileMode Attribute

Valid values for the **TileMode** attribute are None, Tile, FlipX, FlipY, and FlipXY.

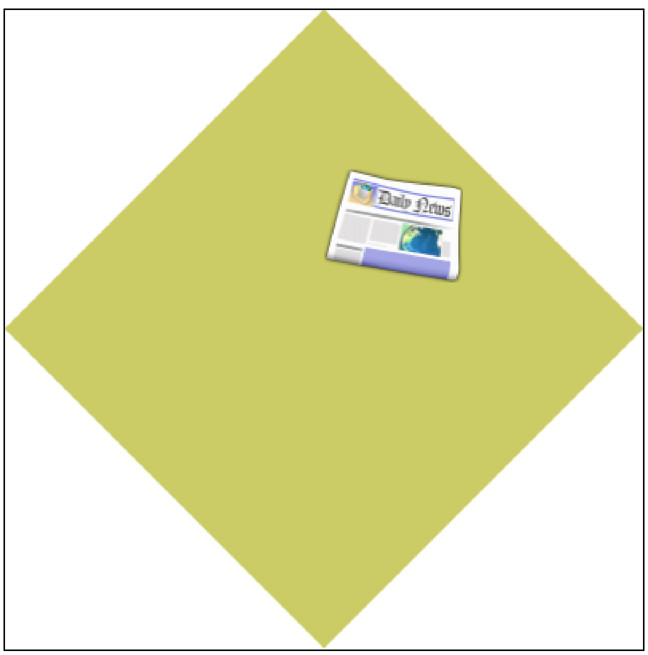
13.4.2.1 None

In this mode, only the single base tile is drawn. The remaining area is left transparent.

Example 13–9. Image brush with TileMode value of None

```
<!-- Draw background diamond to show where fill affects background -->
<Path Fill="#CCCC66" Data="M 200,0 L 400,200 L 200,400 L 0,200 Z" />
<Path Data="M 200,0 L 400,200 L 200,400 L 0,200 Z">
   <Path.Fill>
      <ImageBrush
         ImageSource="newspaper.png"
         Viewbox="0,0,350,284"
         Viewport="200,100,87,71"
         ViewportUnits="Absolute"
         ViewboxUnits="Absolute"
         TileMode="None" />
   </Path.Fill>
</Path>
```

This markup is rendered as follows:

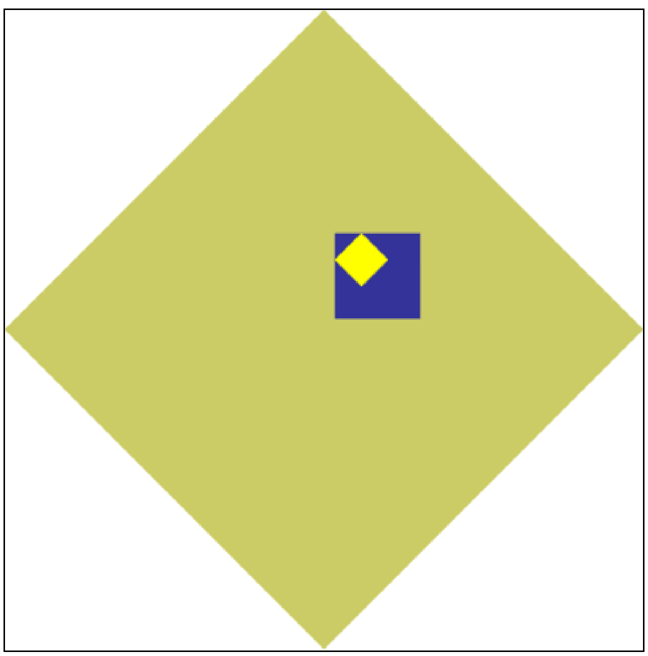


end example]

Example 13–10. Visual brush with TileMode value of None

```
<!-- Draw background diamond to show where fill affects background -->
<Path Fill="#CCCC66" Data="M 200,0 L 400,200 L 200,400 L 0,200 Z" />
<Path Data="M 200,0 L 400,200 L 200,400 L 0,200 Z">
   <Path.Fill>
      <VisualBrush
         Viewbox="0,0,1,1"
         Viewport="200,133,67,67"
         ViewboxUnits="Absolute"
         ViewportUnits="Absolute"
         TileMode="None">
         <VisualBrush.Visual>
            <Canvas>
               <Path Fill="#333399" Data="M 0.1,0.1 L 0.9,0.1 L 0.9,0.9
                  L 0.1,0.9 Z" />
               <Path Fill="#FFFF00" Data="M 0.1,0.35 L 0.35,0.1
                  L 0.6,0.35 L 0.35,0.6 Z" />
            </Canvas>
         </VisualBrush.Visual>
      </VisualBrush>
   </Path.Fill>
</Path>
```

This markup is rendered as follows:



end example]

13.4.2.2 Tile

In this mode, the base tile is drawn and the remaining area is filled by repeating the base tile such that the right edge of each tile abuts the left edge of the next, and the bottom edge of each tile abuts the top edge of the next.

Example 13–11. Image brush with a TileMode value of Tile

```
<!-- Draw background diamond to show where fill affects background -->
<Path Fill="#CCCC66" Data="M 200,0 L 400,200 L 200,400 L 0,200 Z" />
<Path Data="M 200,0 L 400,200 L 200,400 L 0,200 Z">
   <Path.Fill>
      <ImageBrush
         ImageSource="newspaper.png"
         Viewbox="0,0,350,284"
         Viewport="200,100,87,71"
         ViewportUnits="Absolute"
         ViewboxUnits="Absolute"
         TileMode="Tile" />
   </Path.Fill>
</Path>
```

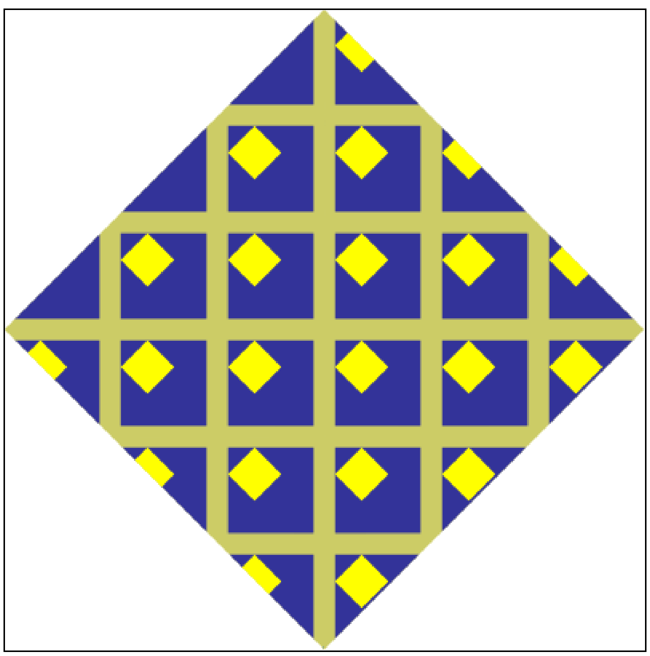
This markup is rendered as follows:

end example]

Example 13–12. Visual brush with a TileMode value of Tile

```
<!-- Draw background diamond to show where fill affects background -->
<Path Fill="#CCCC66" Data="M 200,0 L 400,200 L 200,400 L 0,200 Z" />
<Path Data="M 200,0 L 400,200 L 200,400 L 0,200 Z">
   <Path.Fill>
      <VisualBrush
         Viewbox="0,0,1,1"
         Viewport="200,133,67,67"
         ViewboxUnits="Absolute"
         ViewportUnits="Absolute"
         TileMode="Tile">
         <VisualBrush.Visual>
            <Canvas>
               <Path Fill="#333399" Data="M 0.1,0.1 L 0.9,0.1 L 0.9,0.9
                  L 0.1,0.9 Z" />
               <Path Fill="#FFFF00" Data="M 0.1,0.35 L 0.35,0.1
                  L 0.6,0.35 L 0.35,0.6 Z" />
            </Canvas>
         </VisualBrush.Visual>
      </VisualBrush>
   </Path.Fill>
</Path>
```

This markup is rendered as follows:



end example]

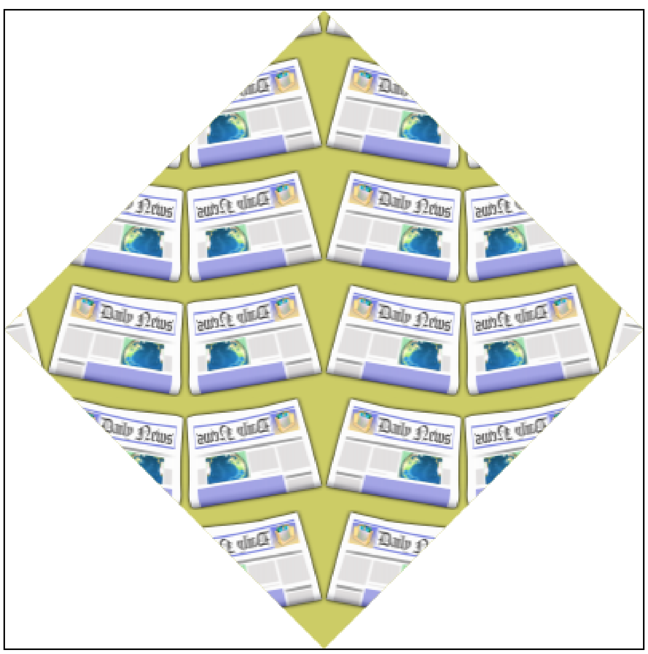
13.4.2.3 FlipX

The tile arrangement is similar to the Tile tile mode, but alternate columns of tiles are flipped horizontally. The base tile is positioned as specified by the viewport. Tiles in the columns to the left and right of this tile are flipped horizontally.

Example 13–13. Image brush with a TileMode value of FlipX

```
<!-- Draw background diamond to show where fill affects background -->
<Path Fill="#CCCC66" Data="M 200,0 L 400,200 L 200,400 L 0,200 Z" />
<Path Data="M 200,0 L 400,200 L 200,400 L 0,200 Z">
   <Path.Fill>
      <ImageBrush
         ImageSource="newspaper.png"
         Viewbox="0,0,350,284"
         Viewport="200,100,87,71"
         ViewportUnits="Absolute"
         ViewboxUnits="Absolute"
         TileMode="FlipX" />
   </Path.Fill>
</Path>
```

This markup is rendered as follows:

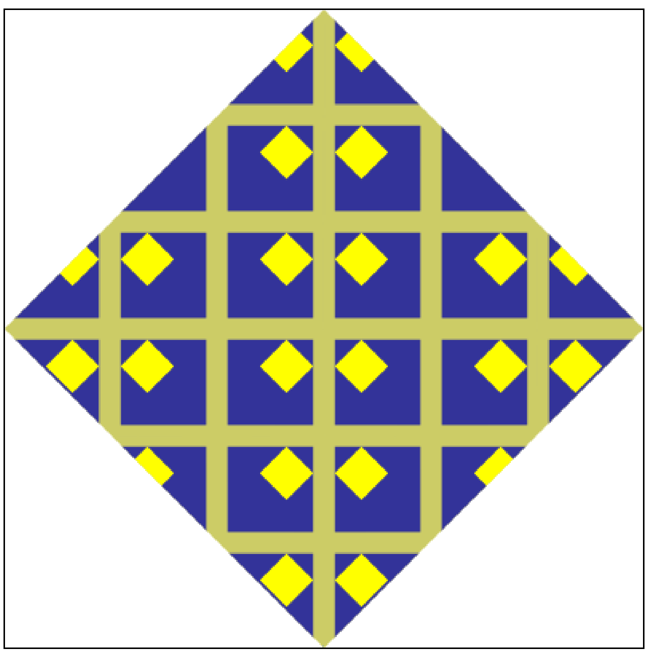


end example]

Example 13–14. Visual brush with a TileMode value of FlipX

```
<!-- Draw background diamond to show where fill affects background -->
<Path Fill="#CCCC66" Data="M 200,0 L 400,200 L 200,400 L 0,200 Z" />
<Path Data="M 200,0 L 400,200 L 200,400 L 0,200 Z">
   <Path.Fill>
      <VisualBrush
         Viewbox="0,0,1,1"
         Viewport="200,133,67,67"
         ViewboxUnits="Absolute"
         ViewportUnits="Absolute"
         TileMode="FlipX">
         <VisualBrush.Visual>
            <Canvas>
               <Path Fill="#333399" Data="M 0.1,0.1 L 0.9,0.1 L 0.9,0.9
                  L 0.1,0.9 Z" />
               <Path Fill="#FFFF00" Data="M 0.1,0.35 L 0.35,0.1
                  L 0.6,0.35 L 0.35,0.6 Z" />
            </Canvas>
         </VisualBrush.Visual>
      </VisualBrush>
   </Path.Fill>
</Path>
```

This markup is rendered as follows:



end example]

13.4.2.4 FlipY

The tile arrangement is similar to the Tile tile mode, but alternate rows of tiles are flipped vertically. The base tile is positioned as specified by the viewport. Rows above and below are flipped vertically.

Example 13–15. Image brush with a TileMode value of FlipY

```
<!-- Draw background diamond to show where fill affects background -->
<Path Fill="#CCCC66" Data="M 200,0 L 400,200 L 200,400 L 0,200 Z" />
<Path Data="M 200,0 L 400,200 L 200,400 L 0,200 Z">
   <Path.Fill>
      <ImageBrush
         ImageSource="newspaper.png"
         Viewbox="0,0,350,284"
         Viewport="200,100,87,71"
         ViewportUnits="Absolute"
         ViewboxUnits="Absolute"
         TileMode="FlipY" />
   </Path.Fill>
</Path>
```

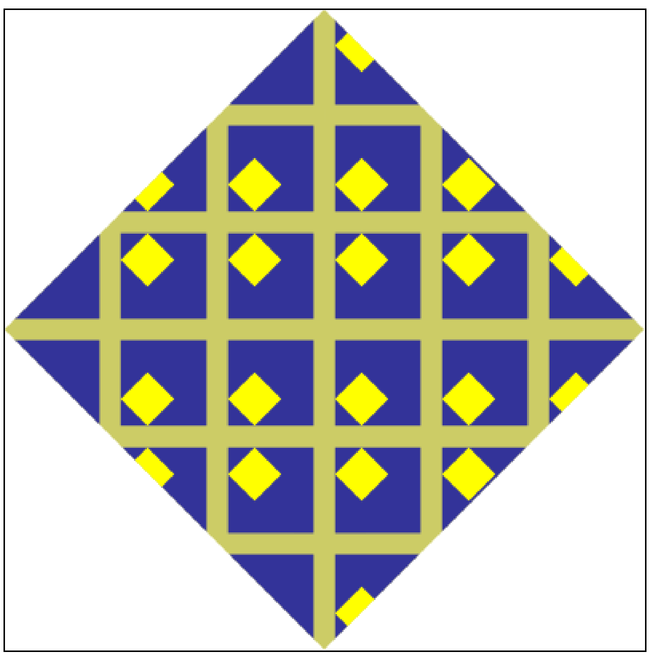
This markup is rendered as follows:

end example]

Example 13–16. Visual Brush with a TileMode value of FlipY

```
<!-- Draw background diamond to show where fill affects background -->
<Path Fill="#CCCC66" Data="M 200,0 L 400,200 L 200,400 L 0,200 Z" />
<Path Data="M 200,0 L 400,200 L 200,400 L 0,200 Z">
   <Path.Fill>
      <VisualBrush
         Viewbox="0,0,1,1"
         Viewport="200,133,67,67"
         ViewboxUnits="Absolute"
         ViewportUnits="Absolute"
         TileMode="FlipY">
         <VisualBrush.Visual>
            <Canvas>
               <Path Fill="#333399" Data="M 0.1,0.1 L 0.9,0.1 L 0.9,0.9
                  L 0.1,0.9 Z" />
               <Path Fill="#FFFF00" Data="M 0.1,0.35 L 0.35,0.1
                  L 0.6,0.35 L 0.35,0.6 Z" />
            </Canvas>
         </VisualBrush.Visual>
      </VisualBrush>
   </Path.Fill>
</Path>
```

This markup is rendered as follows:



end example]

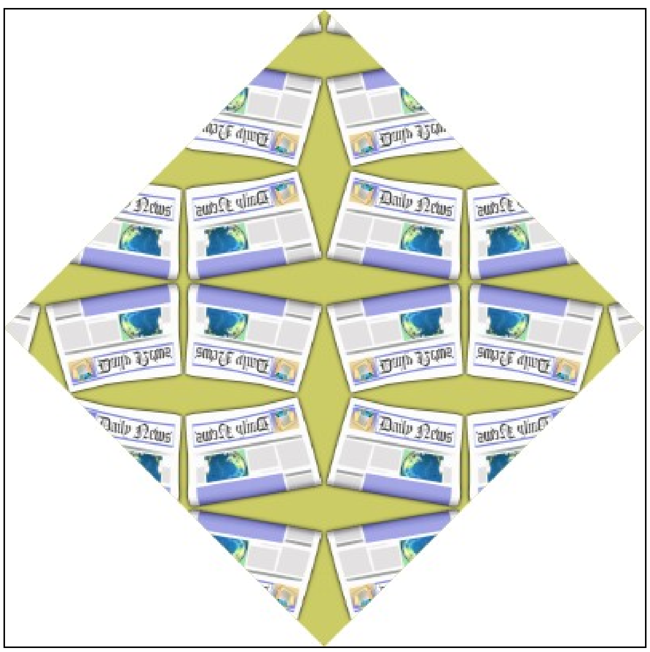
13.4.2.5 FlipXY

The tile arrangement is similar to the Tile tile mode, but alternate columns of tiles are flipped horizontally and alternate rows of tiles are flipped vertically. The base tile is positioned as specified by the viewport.

Example 13–17. Image brush with a TileMode value of FlipXY

```
<!-- Draw background diamond to show where fill affects background -->
<Path Fill="#CCCC66" Data="M 200,0 L 400,200 L 200,400 L 0,200 Z" />
<Path Data="M 200,0 L 400,200 L 200,400 L 0,200 Z">
   <Path.Fill>
      <ImageBrush
         ImageSource="newspaper.png"
         Viewbox="0,0,350,284"
         Viewport="200,100,87,71"
         ViewportUnits="Absolute"
         ViewboxUnits="Absolute"
         TileMode="FlipXY" />
   </Path.Fill>
</Path>
```

This markup is rendered as follows:

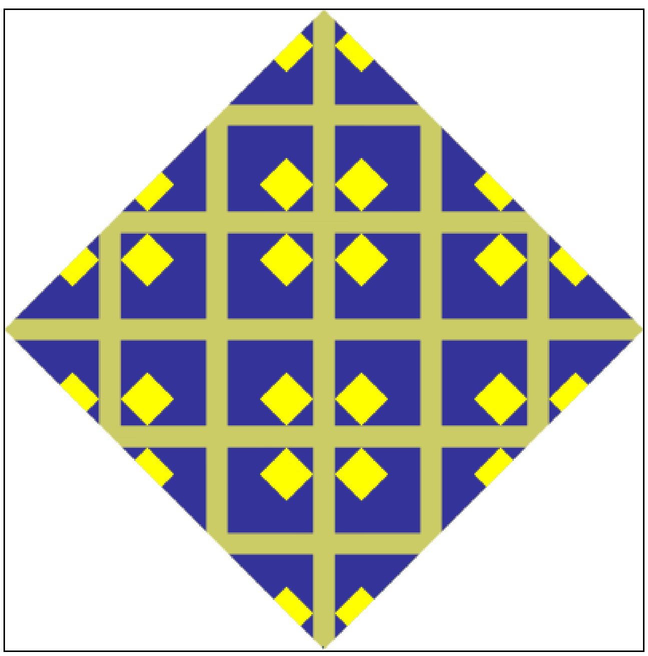


end example]

Example 13–18. Visual brush with a TileMode value of FlipXY

```
<!-- Draw background diamond to show where fill affects background -->
<Path Fill="#CCCC66" Data="M 200,0 L 400,200 L 200,400 L 0,200 Z" />
<Path Data="M 200,0 L 400,200 L 200,400 L 0,200 Z">
   <Path.Fill>
      <VisualBrush
         Viewbox="0,0,1,1"
         Viewport="200,133,67,67"
         ViewboxUnits="Absolute"
         ViewportUnits="Absolute"
         TileMode="FlipXY">
         <VisualBrush.Visual>
            <Canvas>
               <Path Fill="#333399" Data="M 0.1,0.1 L 0.9,0.1 L 0.9,0.9
                  L 0.1,0.9 Z" />
               <Path Fill="#FFFF00" Data="M 0.1,0.35 L 0.35,0.1
                  L 0.6,0.35 L 0.35,0.6 Z" />
            </Canvas>
         </VisualBrush.Visual>
      </VisualBrush>
   </Path.Fill>
</Path>
```

This markup is rendered as follows:

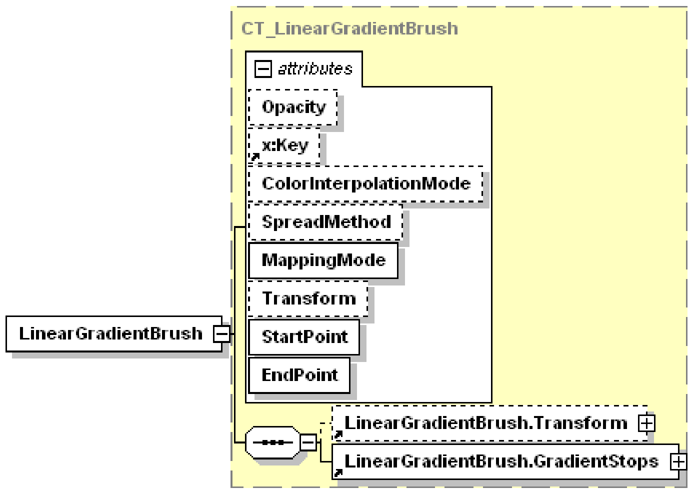


end example]

13.5 <LinearGradientBrush> Element

element **LinearGradientBrush**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
Opacity	ST_ZeroOne		1.0		Defines the uniform transparency of the linear gradient. Values range from 0 (fully transparent) to 1 (fully opaque), inclusive. Values outside of this range are invalid.
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 [M6.5].
ColorInterpolationMode	ST_ClrIntMode		SRgbLinear Interpolation		Specifies the gamma function for color interpolation. The gamma adjustment

					should not be applied to the alpha component, if specified. Valid values are SRgbLinearInterpolation and ScRgbLinearInterpolation.
SpreadMethod	ST_SpreadMethod		Pad		Describes how the brush should fill the content area outside of the primary, initial gradient area. Valid values are Pad, Reflect and Repeat.
MappingMode	ST_MappingMode	required		Absolute	Specifies that the start point and end point are defined in the effective coordinate space (includes the Transform attribute of the brush).
Transform	ST_RscRefMatrix				Describes the matrix transformation applied to the coordinate space of the brush. The Transform property on a brush is concatenated with the current effective render transform to yield an effective render transform local to the brush. The start point and end point are transformed using the local effective render transform.
StartPoint	ST_Point	required			Specifies the starting point of the linear gradient.
EndPoint	ST_Point	required			Specifies the end point of the linear gradient. The linear gradient brush interpolates the colors from the start point to the end point, where the start point represents an offset of

						0, and the EndPoint represents an offset of 1. The Offset attribute value specified in a GradientStop element relates to the 0 and 1 offsets defined by the start point and end point.
annotation	Fills a region with a linear gradient.					

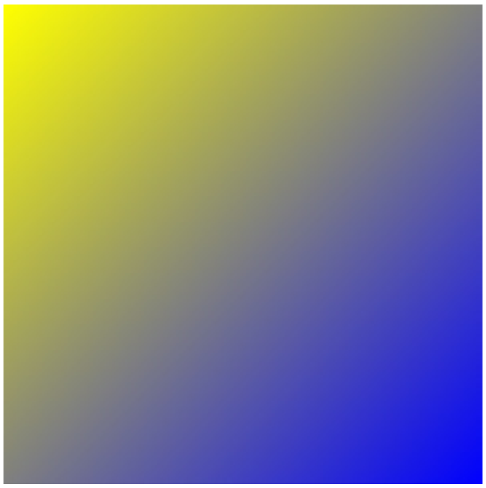
The <LinearGradientBrush> element is used to specify a linear gradient brush along a vector. For details about computing a linear gradient, see Section18.3, “Gradient Computations,” on page 272.

Example 13–19. <LinearGradientBrush> usage

The following markup describes a page with a rectangular path that is filled with a linear gradient:

```
<Path>
   <Path.Fill>
      <LinearGradientBrush
         MappingMode="Absolute"
         StartPoint="0,0"
         EndPoint="300,300">
         <LinearGradientBrush.GradientStops>
            <GradientStop Color="#FFFF00" Offset="0" />
            <GradientStop Color="#0000FF" Offset="1" />
         </LinearGradientBrush.GradientStops>
      </LinearGradientBrush>
   </Path.Fill>
   <Path.Data>
      <PathGeometry>
         <PathFigure StartPoint="0,0">
            <PolyLineSegment Points="300,0 300,300 0,300" />
         </PathFigure>
      </PathGeometry>
   </Path.Data>
</Path>
```

This markup is rendered as follows:



end example]

13.5.1 SpreadMethod Attribute

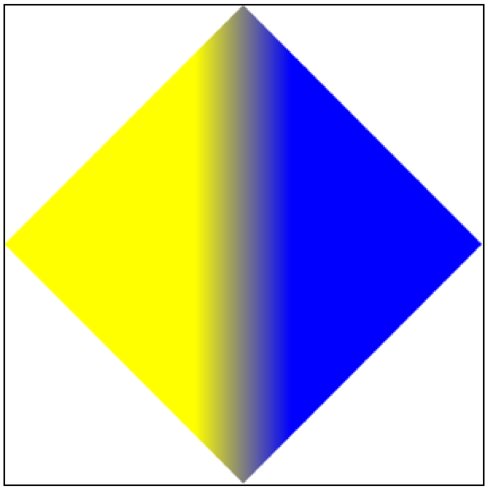
The **SpreadMethod** attribute describes the fill for areas beyond the start point and end point of the linear gradient brush. Valid values are Pad, Reflect, and Repeat.

Example 13–20. Linear gradient brush with a SpreadMethod value of Pad

In this method, the first color and the last color are used to fill the remaining fill area at the beginning and end.

```
<Path Data="M 150,0 L 300,150 L 150,300 L 0,150 Z">
   <Path.Fill>
      <LinearGradientBrush
         MappingMode="Absolute"
         StartPoint="120,0"
         EndPoint="180,0"
         SpreadMethod="Pad">
         <LinearGradientBrush.GradientStops>
            <GradientStop Color="#FFFF00" Offset="0.0" />
            <GradientStop Color="#0000FF" Offset="1.0" />
         </LinearGradientBrush.GradientStops>
      </LinearGradientBrush>
   </Path.Fill>
</Path>
```

This markup is rendered as follows:



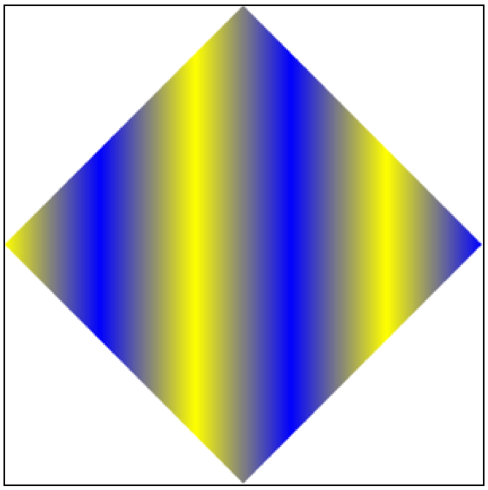
end example]

Example 13–21. Linear gradient brush with a SpreadMethod value of Reflect

In this method, the gradient stops are replayed in reverse order repeatedly to cover the fill area.

```
<Path Data="M 150,0 L 300,150 L 150,300 L 0,150 Z">
   <Path.Fill>
      <LinearGradientBrush
         MappingMode="Absolute"
         StartPoint="120,0"
         EndPoint="180,0"
         SpreadMethod="Reflect">
         <LinearGradientBrush.GradientStops>
            <GradientStop Color="#FFFF00" Offset="0.0" />
            <GradientStop Color="#0000FF" Offset="1.0" />
         </LinearGradientBrush.GradientStops>
      </LinearGradientBrush>
   </Path.Fill>
</Path>
```

This markup is rendered as follows:



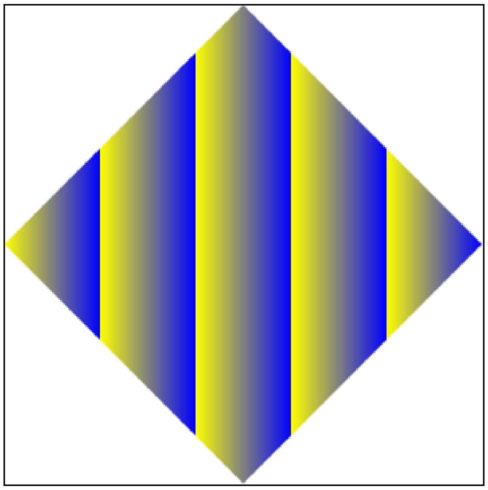
end example]

Example 13–22. Linear gradient brush with a SpreadMethod value of Repeat

In this method, the gradient stops are repeated in order until the fill area is covered.

```
<Path Data="M 150,0 L 300,150 L 150,300 L 0,150 Z">
   <Path.Fill>
      <LinearGradientBrush
         MappingMode="Absolute"
         StartPoint="120,0"
         EndPoint="180,0"
         SpreadMethod="Repeat">
         <LinearGradientBrush.GradientStops>
            <GradientStop Color="#FFFF00" Offset="0.0" />
            <GradientStop Color="#0000FF" Offset="1.0" />
         </LinearGradientBrush.GradientStops>
      </LinearGradientBrush>
   </Path.Fill>
</Path>
```

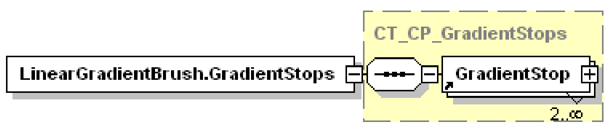
This markup is rendered as follows:



end example]

13.5.2 <LinearGradientBrush.GradientStops> Element

element **LinearGradientBrush.GradientStops**

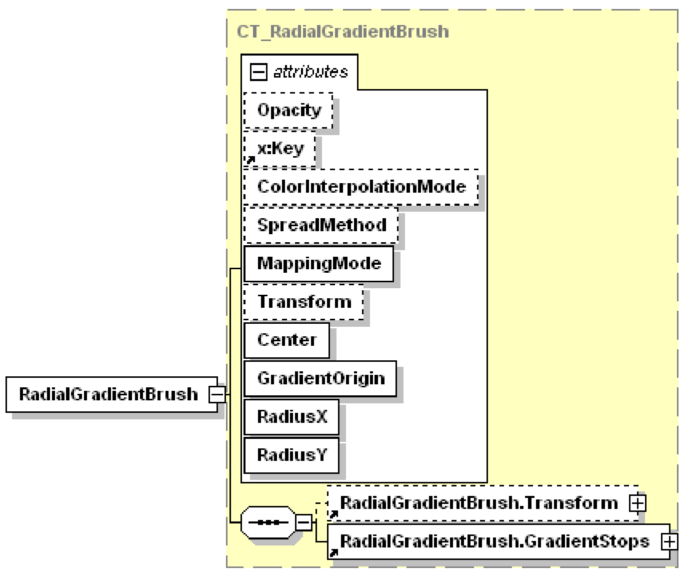
diagram	
annotation	Holds a sequence of GradientStop elements.

The <LinearGradientBrush.GradientStops> property element specifies a collection of gradient stops that comprise the linear gradient. For more information, see §13.7, "<GradientStop> Element," on page 166.

13.6 <RadialGradientBrush> Element

element **RadialGradientBrush**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
Opacity	ST_ZeroOne		1.0		Defines the uniform transparency of the radial gradient. Values range from 0 (fully transparent) to 1 (fully opaque), inclusive. Values outside of this range are invalid.
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 [M6.6].
ColorInterpolationMode	ST_ClrIntMode		SRgbLinear		Specifies the gamma

			Interpolation		function for color interpolation for sRGB colors. The gamma adjustment should not be applied to the alpha component, if specified. Valid values are SRgbLinearInterpolation and ScRgbLinearInterpolation.
SpreadMethod	ST_SpreadMethod		Pad		Describes how the brush should fill the content area outside of the primary, initial gradient area. Valid values are Pad, Reflect and Repeat.
MappingMode	ST_MappingMode	required		Absolute	Specifies that center, x radius, and y radius are defined in the effective coordinate space (includes the Transform attribute of the brush).
Transform	ST_RscRefMatrix				Describes the matrix transformation applied to the coordinate space of the brush. The Transform property is concatenated with the current effective render transform to yield an effective render transform local to the brush. The ellipse defined by the center, gradient origin, x radius, and y radius values is transformed using the local effective render transform.
Center	ST_Point	required			Specifies the center point of the radial gradient (that is, the center of the ellipse). The radial gradient brush interpolates the colors from the gradient origin to the circumference of the

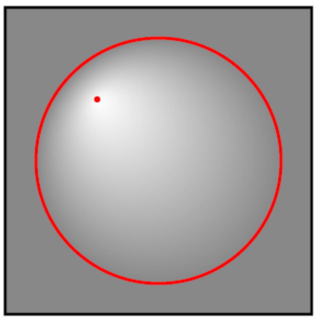
						ellipse. The circumference is determined by the center and the radii.
	GradientOrigin	ST_Point	required			Specifies the origin point of the radial gradient.
	RadiusX	ST_GEZero	required			Specifies the radius in the x dimension of the ellipse which defines the radial gradient.
	RadiusY	ST_GEZero	required			Specifies the radius in the y dimension of the ellipse which defines the radial gradient.
annotation	Fills a region with a radial gradient.					

Radial gradient brushes are similar to linear gradient brushes. However, whereas a linear gradient brush has a start point and end point to define the gradient vector, a radial gradient brush has an ellipse (defined by the center, *x* radius, and *y* radius) and a gradient origin. The ellipse defines the end point of the gradient. In other words, a gradient stop with an offset at 1.0 defines the color at the circumference of the ellipse. The gradient origin defines the center of the gradient. A gradient stop with an offset at 0.0 defines the color at the gradient origin.

For details about computing a radial gradient, see §18.3.3, "Radial Gradients," on page 276.

Example 13–23. A radial gradient brush

The following figure is a radial gradient that transitions from white to gray. The outside ellipse represents the gradient ellipse while the dot denotes the gradient origin. This gradient has a **SpreadMethod** value of Pad. *end example*]

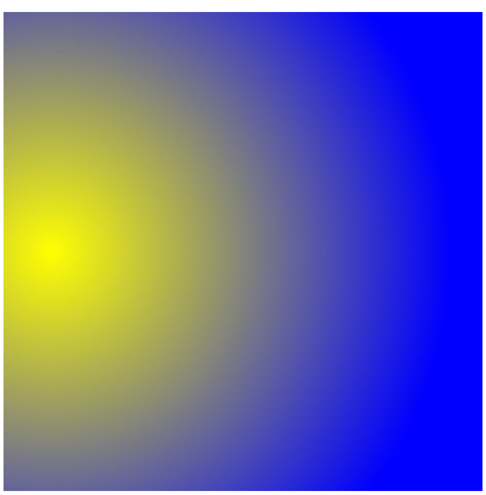


Example 13–24. RadialGradientBrush usage

The following markup describes a page with a rectangular path that is filled with a radial gradient:

```
<Path>
   <Path.Fill>
      <RadialGradientBrush
         MappingMode="Absolute"
         Center="30,150"
         GradientOrigin="30,150"
         RadiusX="250"
         RadiusY="250">

<RadialGradientBrush.GradientStops>
            <GradientStop Color="#FFFF00"
Offset="0" />
            <GradientStop Color="#0000FF" Offset="1" />
         </RadialGradientBrush.GradientStops>
      </RadialGradientBrush>
   </Path.Fill>
   <Path.Data>
      <PathGeometry>
         <PathFigure StartPoint="0,0" IsClosed="true">
            <PolyLineSegment Points="300,0 300,300 0,300" />
         </PathFigure>
      </PathGeometry>
   </Path.Data>
</Path>
```

This markup is rendered as follows:



end example]

13.6.1 SpreadMethod Attribute

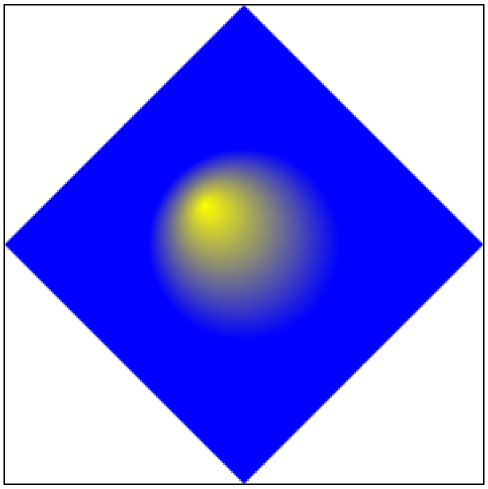
The **SpreadMethod** attribute describes the fill of areas beyond the ellipse described by the center, *x* radius, and *y* radius of the radial gradient brush. Valid values are Pad, Reflect, and Repeat.

Example 13–25. Radial gradient brush with a SpreadMethod value of Pad

In the following markup, the last color is used to cover the fill area outside the ellipse.

```
<Path Data="M 150,0 L 300,150 L 150,300 L 0,150 Z">
   <Path.Fill>
      <RadialGradientBrush
         MappingMode="Absolute"
         Center="150,150"
         GradientOrigin="125,125"
         RadiusX="60"
         RadiusY="60"
         SpreadMethod="Pad">
         <RadialGradientBrush.GradientStops>
            <GradientStop Color="#FFFF00" Offset="0.0" />
            <GradientStop Color="#0000FF" Offset="1.0" />
         </RadialGradientBrush.GradientStops>
      </RadialGradientBrush>
   </Path.Fill>
</Path>
```

This markup is rendered as follows:



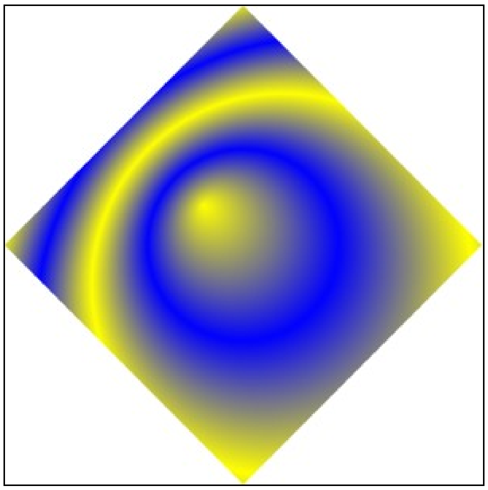
end example]

Example 13–26. Radial gradient brush with a SpreadMethod value of Reflect

In the following markup, the gradient stops are replayed in reverse order repeatedly to cover the fill area.

```
<Path Data="M 150,0 L 300,150 L 150,300 L 0,150 Z">
   <Path.Fill>
      <RadialGradientBrush
         MappingMode="Absolute"
         Center="150,150"
         GradientOrigin="125,125"
         RadiusX="60"
         RadiusY="60"
         SpreadMethod="Reflect">
         <RadialGradientBrush.GradientStops>
            <GradientStop Color="#FFFF00" Offset="0.0" />
            <GradientStop Color="#0000FF" Offset="1.0" />
         </RadialGradientBrush.GradientStops>
      </RadialGradientBrush>
   </Path.Fill>
</Path>
```

This markup is rendered as follows:



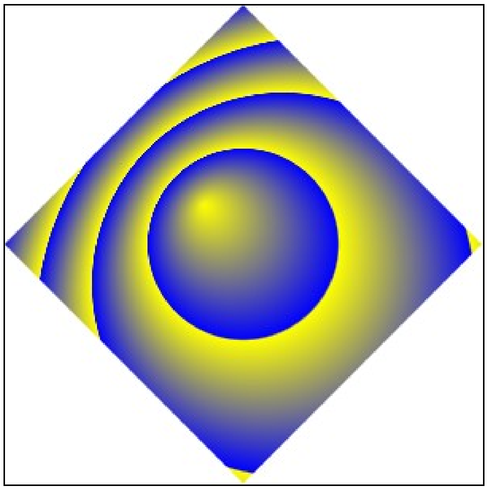
end example]

Example 13–27. Radial gradient brush with a SpreadMethod value of Repeat

In the following markup, the gradient stops are repeated in order until the fill area is covered.

```
<Path Data="M 150,0 L 300,150 L 150,300 L 0,150 Z">
   <Path.Fill>
      <RadialGradientBrush
         MappingMode="Absolute"
         Center="150,150"
         GradientOrigin="125,125"
         RadiusX="60"
         RadiusY="60"
         SpreadMethod="Repeat">
         <RadialGradientBrush.GradientStops>
            <GradientStop Color="#FFFF00" Offset="0.0" />
            <GradientStop Color="#0000FF" Offset="1.0" />
         </RadialGradientBrush.GradientStops>
      </RadialGradientBrush>
   </Path.Fill>
</Path>
```

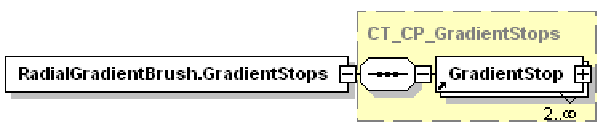
This markup is rendered as follows:



end example]

13.6.2 <RadialGradientBrush.GradientStops> Element

element **RadialGradientBrush.GradientStops**

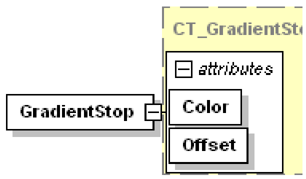
diagram	
annotation	Holds a sequence of <GradientStop> elements.

The <RadialGradientBrush.GradientStops> property element specifies a collection of gradient stops that comprise the radial gradient. For more information, see §13.7, "<GradientStop> Element," on page 166.

13.7 <GradientStop> Element

element **GradientStop**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
Color	ST_Color	required			Specifies the gradient stop color. An sRGB color value specified as a 6-digit hexadecimal number (#RRGGBB) or an extended color.
Offset	ST_Double	required			Specifies the gradient offset. The offset indicates a point along the progression of the gradient at which a color is specified. Colors between gradient offsets in the progression are interpolated.

annotation: Indicates a location and range of color progression for rendering a gradient.

The <GradientStop> element is used by both the <LinearGradientBrush> and <RadialGradientBrush> elements to define the location and range of color progression for rendering a gradient.

For linear gradient brushes, the offset value of 0.0 is mapped to the start point of the gradient, and the offset value of 1.0 is mapped to the end point. Intermediate offset values are interpolated between these two points to determine their location.

For radial gradient brushes, the offset value of 0.0 is mapped to the gradient origin location. The offset value of 1.0 is mapped to the circumference of the ellipse as determined by the center, *x* radius, and *y* radius. Offsets between 0.0 and 1.0 are positioned at a location interpolated between these points.

For full details of rendering of gradient brushes, including handling of offsets, please see §18.3, “Gradient Computations” on page 272.

13.8 Using a Brush as an Opacity Mask

Each pixel carries an alpha value ranging from 0.0 (fully transparent) to 1.0 (fully opaque). The alpha value is used when blending elements to achieve the visual effect of transparency. Each element can have an **Opacity** attribute by which the alpha value of each pixel is multiplied uniformly.

The OpacityMask property also allows the specification of per-pixel opacity, which controls how rendered content is blended with its destination. The opacity specified by the opacity mask is combined multiplicatively with any opacity that may already be present in the alpha channel of the contents. The per-pixel opacity specified by the opacity mask is determined by the alpha channel of each pixel in the mask. The color data is ignored.

The alpha value of the area not marked by the brush is 0.0. The required computations for transparently blending two elements when rendering, also known as *alpha blending*, are described in §18.4, “Opacity Computations” on page 279.

An opacity mask always has a brush child element.

Example 13–28. Opacity mask with linear gradient

The following markup illustrates how an opacity mask is used to create a fade effect on a glyph. The opacity mask is a linear gradient that fades from opaque black to transparent black.

```
<FixedPage Height="1056" Width="816" xml:lang="en-US">
   <Glyphs
      OriginX="25"
      OriginY="50"
      UnicodeString="This is a fading text example."
      FontUri="../Resources/Fonts/Times.TTF"
      FontRenderingEmSize="32">
      <Glyphs.OpacityMask>
         <LinearGradientBrush
            StartPoint="25,0"
            EndPoint="450,0"
            MappingMode="Absolute">
            <LinearGradientBrush.GradientStops>
               <GradientStop Color="#FF000000" Offset="0" />
               <GradientStop Color="#00000000" Offset="1" />
            </LinearGradientBrush.GradientStops>
         </LinearGradientBrush>
      </Glyphs.OpacityMask>
      <Glyphs.Fill>
         <SolidColorBrush Color="#000000" />
      </Glyphs.Fill>
   </Glyphs>
</FixedPage>
```

This markup is rendered as follows:

end example]

Example 13–29. Opacity mask with radial gradient

In the following markup, the opacity mask is a radial gradient:

```
<FixedPage Width="816" Height="1056" xml:lang="en-US">
   <Path>
      <Path.OpacityMask>
         <RadialGradientBrush
            MappingMode="Absolute"
            Center="200,300"
            GradientOrigin="200,300"
            RadiusX="200"
            RadiusY="300">
            <RadialGradientBrush.GradientStops>
               <GradientStop Color="#FF000000" Offset="0" />
               <GradientStop Color="#20000000" Offset="1" />
            </RadialGradientBrush.GradientStops>
         </RadialGradientBrush>
      </Path.OpacityMask>
      <Path.Fill>
         <ImageBrush
            Viewbox="0,0,400,600"
            ViewboxUnits="Absolute"
            Viewport="0,0,400,600"
            ViewportUnits="Absolute"
            TileMode="None"
            ImageSource="images/jpeg3.jpg" />
      </Path.Fill>
      <Path.Data>
         <PathGeometry>
            <PathFigure StartPoint="0,0" IsClosed="true">
               <PolyLineSegment Points="400,0 400,600 0,600" />
            </PathFigure>
         </PathGeometry>
      </Path.Data>
   </Path>
</FixedPage>
```

This markup is rendered as follows:

end example]

14. Common Properties

Several XPS Document elements share property attributes and elements as summarized in Table 14–1 and Table 14–2 and detailed in the following sections. Other than the Name, FixedPage.NavigateUri, and xml:lang attributes, these properties compose their results from parent to child, as described in §18.4.1, "Pre-Multiplied Alpha and Superluminous Colors," on page 281.

Table 14–1. Common property attributes

Name	Applies to	Description
Clip	<Canvas> <Glyphs> <Path>	Restricts the region to which a brush can be applied.
Opacity	<Canvas> <Glyphs> <ImageBrush> <LinearGradientBrush> <Path> <RadialGradientBrush> <SolidColorBrush> <VisualBrush>	Defines the uniform transparency of the element.
OpacityMask	<Canvas> <Glyphs> <Path>	Specifies a mask of alpha values.
RenderTransform	<Canvas> <Glyphs> <Path>	Establishes a new coordinate space through the use of an affine matrix transformation. For more information, see §14.4, "Positioning Content," on page 185.
Transform	<ImageBrush> <LinearGradientBrush> <PathGeometry> <RadialGradientBrush> <VisualBrush>	Establishes a new coordinate space through the use of an affine matrix transformation. Geometry transformations are applied before brushes. The results are concatenated with any containing effective render transformation specification.
Name	<Canvas> <FixedPage> <Glyphs> <Path>	Defines a hyperlink target or identifies an element uniquely for document structure markup to reference. For more information, see §16.2, "Hyperlinks," on page 243.

FixedPage.NavigateUri	<Canvas> <Glyphs> <Path>	Defines a hyperlink source. For more information, see §16.2, "Hyperlinks," on page 243.
xml:lang	<Canvas> <FixedPage> <Glyphs> <Path>	Specifies a language.

Table 14–2. Common property elements

Name	**Description**
<Canvas.Resources> <FixedPage.Resources>	Contains elements that may be reused by reference throughout the markup of the <FixedPage> or <Canvas> child or descendant elements.
<Canvas.Clip> <Glyphs.Clip> <Path.Clip>	Restricts the region to which a brush can be applied.
<Canvas.RenderTransform> <Glyphs.RenderTransform> <Path.RenderTransform>	Establishes a new coordinate space through the use of an affine matrix transformation. For more information, see §14.4, "Positioning Content," on page 185.
<ImageBrush.Transform> <LinearGradientBrush.Transform> <PathGeometry.Transform> <RadialGradientBrush.Transform> <VisualBrush.Transform>	Establishes a new effective coordinate space through the use of an affine matrix transformation. Path geometry transformations (<PathGeometry.Transform>) are applied before brushes. The results are concatenated with any containing effective render transformation.
<Canvas.OpacityMask> <Glyphs.OpacityMask> <Path.OpacityMask>	Specifies a mask of alpha values that is applied in the same fashion as the Opacity attribute, but allows different alpha values on a pixel-by-pixel basis.

14.1 Opacity

The **Opacity** property attribute is used to transparently blend the current element with previously specified elements, also known as alpha blending. The opacity value MUST fall within the 0 (fully transparent) to 1 (fully opaque) range, inclusive [M2.72].

For more information, see §18.4, "Opacity Computations," on page 279.

14.2 Resources and Resource References

Fixed page markup supports the concept of resources. A *resource* is a reusable property value that is expressed in markup, identified by a key, and stored in a *resource dictionary*. In general, any property value that can be expressed using property element syntax may be held in a resource dictionary.

Each resource in a resource dictionary has a key. Any property that specifies its value by referencing a resource key in a resource dictionary is called a *resource reference*.

The <Canvas> and <FixedPage> elements can carry a resource dictionary. A resource dictionary is expressed in markup by the <FixedPage.Resources> or <Canvas.Resources> property element. Individual resource values MUST be specified within a resource dictionary [M7.1].

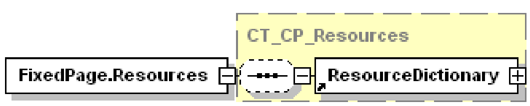
The <Canvas.Resources> or <FixedPage.Resources> property elements MUST precede any property elements of the <Canvas> or <FixedPage> elements [M2.72]. Likewise, they MUST precede any path, glyphs, or canvas children of the <Canvas> or <FixedPage> elements [M2.72].

Alternatively, resource dictionaries MAY be specified in separate parts and referenced from within the <FixedPage.Resources> or <Canvas.Resources> property element [O7.1]. Such a *remote resource dictionary* can be shared across multiple pages. [*Example*: By defining a brush in a remote resource dictionary, graphical elements that are common to multiple pages can be reused. *end example*]

The <Path>, <Glyphs>, and <Canvas> elements can appear as a resource definition solely for the purpose of using these elements in the **Visual** attribute of a <VisualBrush> element. Brushes and geometries appear in resource dictionaries far more frequently.

14.2.1 <FixedPage.Resources> Element

element **FixedPage.Resources**

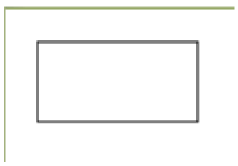
diagram	
annotation	Contains the resource dictionary for the <FixedPage> element.

Example 14–1. <FixedPage.Resources> usage

```
<FixedPage Width="816" Height="1056" xml:lang="en-US"
   xmlns="http://schemas.microsoft.com/xps/2005/06"
   xmlns:x="http://schemas.microsoft.com/xps/2005/06/resourcedictionary-
      key">
   <FixedPage.Resources>
      <ResourceDictionary>
         <PathGeometry x:Key="Rectangle">
            <PathFigure StartPoint="20,20" IsClosed="true">
               <PolyLineSegment Points="120,20 120,70 20,70" />
            </PathFigure>
         </PathGeometry>
      </ResourceDictionary>
   </FixedPage.Resources>
   <Path Stroke="#000000"
      StrokeThickness="1"
```

```
      Data="{StaticResource Rectangle}" />
</FixedPage>
```

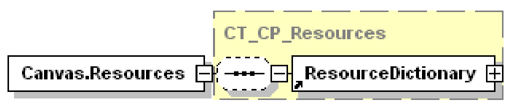
This markup is rendered as follows:



end example]

14.2.2 <Canvas.Resources> Element

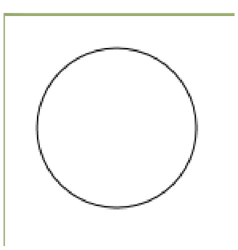
element **Canvas.Resources**

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

Example 14–2. <Canvas.Resources> usage

```
<Canvas
   xmlns:x="http://schemas.microsoft.com/xps/2005/06/resourcedictionary-
      key">
   <Canvas.Resources>
      <ResourceDictionary>
         <PathGeometry x:Key="Circle">
            <PathFigure StartPoint="20,70">
               <ArcSegment
                  Point="120,70"
                  Size="50,50"
                  RotationAngle="0"
                  IsLargeArc="true"
                  SweepDirection="Clockwise" />
               <ArcSegment
                  Point="20,70"
                  Size="50,50"
                  RotationAngle="0"
                  IsLargeArc="true"
                  SweepDirection="Clockwise" />
            </PathFigure>
         </PathGeometry>
      </ResourceDictionary>
   </Canvas.Resources>
   <Path Stroke="#000000"
      StrokeThickness="1"
      Data="{StaticResource Circle}" />
</Canvas>
```

This markup is rendered as follows:

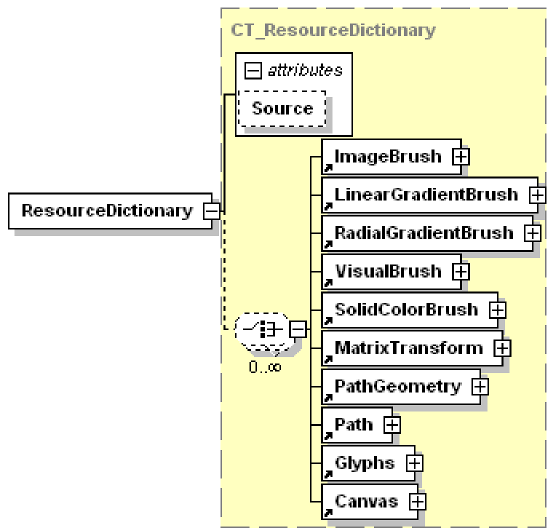


end example]

14.2.3 <ResourceDictionary> Element

element **ResourceDictionary**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
Source	xs:anyURI				Specifies the URI of a part containing markup for a resource dictionary. The URI MUST refer to a part in the package [M2.1].

annotation

Defines a set of reusable resource definitions that can be used as property values in the fixed page markup.

The <FixedPage.Resources> and <Canvas.Resources> property elements contain exactly one <ResourceDictionary> element. A resource dictionary contains *resource definition* element entries. Each resource definition has a key specified in the **x:Key** attribute that is unique within the scope of the resource dictionary. The **x:Key** attribute is included in the Resource Dictionary namespace specified in H, "Standard Namespaces and Content Types" on page 405.

Resource dictionaries can be declared inline inside a <FixedPage.Resources> or <Canvas.Resources> element, or they MAY be defined in a separate part and referenced by a <ResourceDictionary> element inside a <FixedPage.Resources> or <Canvas.Resources> element [O7.1]. This allows resource dictionaries to be shared across parts. [*Example*: A single resource dictionary can be used by every fixed page in the XPS Document. *end example*] See §14.2.3.1, "Remote Resource Dictionaries," on page 177 for more details.

A resource definition MAY reference another resource defined previously in the same resource dictionary [O7.2]. 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 [O7.3].

Namespace prefixes in resource definitions MUST apply in the context of the definition, rather than in the context of the resource reference [M7.2]. An **xml:lang** attribute within a resource definition MUST be interpreted in the context of the resource reference, not the resource definition [M7.3].

Example 14–3. Resource dictionary markup

The following markup defines two geometries, one for a rectangle, and the other for a circle:

```
<Canvas
   xmlns:x="http://schemas.microsoft.com/xps/2005/06/resourcedictionary-
      key">
   <Canvas.Resources>
      <ResourceDictionary>
         <PathGeometry x:Key="Rectangle">
            <PathFigure StartPoint="20,20" IsClosed="true">
               <PolyLineSegment Points="120,20 120,70 20,70" />
            </PathFigure>
         </PathGeometry>
         <PathGeometry x:Key="Circle">
            <PathFigure StartPoint="20,70">
               <ArcSegment
                  Point="120,70"
                  Size="50,50"
                  RotationAngle="0"
                  IsLargeArc="true"
                  SweepDirection="Clockwise" />
               <ArcSegment
                  Point="20,70"
                  Size="50,50"
                  RotationAngle="0"
                  IsLargeArc="true"
                  SweepDirection="Clockwise" />
            </PathFigure>
         </PathGeometry>
      </ResourceDictionary>
   </Canvas.Resources>
   <Path Data="{StaticResource Rectangle}">
      <Path.Fill>
         <SolidColorBrush Color="#FF0000" />
      </Path.Fill>
   </Path>
</Canvas>
```

end example]

14.2.3.1 Remote Resource Dictionaries

A resource dictionary MAY be defined in a separate part [O7.1]. This is referred to as a *remote resource dictionary*. A remote resource dictionary MUST follow the requirements above that apply to all resource dictionaries [M7.4]. A remote resource dictionary MUST NOT contain any resource definition children that reference another remote resource dictionary [M7.5].

The <FixedPage.Resources> and <Canvas.Resources> property elements include a remote resource dictionary via reference, using the **Source** attribute of the <ResourceDictionary> element.

A <ResourceDictionary> element that specifies a remote resource dictionary in its **Source** attribute MUST NOT contain any resource definition children [M7.6]. <FixedPage.Resources> and <Canvas.Resources> elements that include a remote resource dictionary MUST include exactly one <ResourceDictionary> element [M2.72].

A remote Resource Dictionary part MUST be added as a Required Resource relationship from the FixedPage part that references it [M2.10]. In addition, producers MUST add each resource such as fonts or images referenced in the Resource Dictionary part as a Required Resource relationship from the FixedPage part (*not* the Resource Dictionary part) to the indirectly required resource, even if the particular fixed page does not reference the resource [M2.10]. For more information, see §H.3, "Relationship Types," on page 407.

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 [M7.7].

Example 14–4. A remote resource dictionary and reference

The following markup defines a resource dictionary that contains two geometries, one for a rectangle and the other for a circle:

```
<!-- Contents of /resource.xaml -->
<ResourceDictionary xmlns="http://schemas.microsoft.com/xps/2005/06"
   xmlns:x="http://schemas.microsoft.com/xps/2005/06/resourcedictionary-
      key">
   <PathGeometry x:Key="Rectangle">
      <PathFigure StartPoint="20,20" IsClosed="true">
         <PolyLineSegment Points="120,20 120,70 20,70" />
      </PathFigure>
   </PathGeometry>
   <PathGeometry x:Key="Circle">
      <PathFigure StartPoint="20,70">
         <ArcSegment
            Point="120,70"
            Size="50,50"
            RotationAngle="0"
            IsLargeArc="true"
            SweepDirection="Clockwise" />
         <ArcSegment
            Point="20,70"
            Size="50,50"
            RotationAngle="0"
            IsLargeArc="true"
            SweepDirection="Clockwise" />
      </PathFigure>
   </PathGeometry>
</ResourceDictionary>
```

The following markup references the previously defined resource dictionary:

```
<Canvas
   xmlns:x="http://schemas.microsoft.com/xps/2005/06/resourcedictionary-
      key">
```

```
    <Canvas.Resources>
       <ResourceDictionary Source="/resource.xaml"/>
    </Canvas.Resources>
    <Path Data="{StaticResource Rectangle}">
       <Path.Fill>
          <SolidColorBrush Color="#FF0000" />
       </Path.Fill>
    </Path>
</Canvas>
```

end example]

14.2.4 Resource References

To set a property value to a defined resource, use the form:

```
{StaticResource key}
```

Where *key* is the same string specified with **x:Key** in the resource definition.

The context of the resource reference determines how defined resources are rendered (such as the transformation matrix to be applied). Specifically, the effective coordinate space for rendering the referenced resource is a composition of the effective coordinate space of the referring element plus any Transform or RenderTransform properties included in the resource definition itself.

It is considered an error if a static resource reference cannot be resolved, or if it *can* be resolved but the resource type does not match the usage at the location of reference.

Example 14–5. Using a resource reference to fill a brush

In the following markup, the rectangular region defined by the geometry specified in the dictionary is filled by a solid color brush:

```
<Canvas
   xmlns:x="http://schemas.microsoft.com/xps/2005/06/resourcedictionary-
      key">
   <Canvas.Resources>
      <ResourceDictionary>
         <PathGeometry x:Key="Rectangle">
            <PathFigure StartPoint="20,20" IsClosed="true">
               <PolyLineSegment Points="120,20 120,70 20,70" />
            </PathFigure>
         </PathGeometry>
      </ResourceDictionary>
   </Canvas.Resources>
   <Path Data="{StaticResource Rectangle}">
      <Path.Fill>
         <SolidColorBrush Color="#FF0000" />
      </Path.Fill>
   </Path>
</Canvas>
```

end example]

14.2.5 Scoping Rules for Resolving Resource References

The value of the **x:Key** attribute MUST be unique within the resource dictionary [M2.72]. However, the resource dictionary of a <Canvas> element MAY re-use an **x:Key** value defined in the resource dictionary of a parent or ancestor <Canvas> or <FixedPage> element [O7.5]. Resource references are resolved from the innermost to the outermost resource dictionary.

A resource definition MAY reference a previously defined resource with the same name that is defined in an ancestor resource dictionary [O7.6]; the reference MUST be resolved before the redefined resource is added to the dictionary [M7.8].

A resource definition MAY reference another resource defined prior to the point of reference, including a resource previously defined within the same resource dictionary [O7.2]. 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 [M7.9].

To find a resource, the nearest parent or ancestor canvas or fixed page is searched. If the desired name is not defined in the initially searched resource dictionary, then the next-nearest parent or ancestor canvas or fixed page is searched. An error occurs if the search has continued to the root <FixedPage> element and a specified resource has not been found. This search occurs only within the containing FixedPage part.

Example 14–6. Using scoping rules

```
<FixedPage
   xmlns="http://schemas.microsoft.com/xps/2005/06"
   xmlns:x="http://schemas.microsoft.com/xps/2005/06/resourcedictionary-
      key"
   Height="1056" Width="816" xml:lang="en-US">
   <FixedPage.Resources>
      <ResourceDictionary>
         <SolidColorBrush x:Key="FavoriteColorFill" Color="#808080" />
      </ResourceDictionary>
   </FixedPage.Resources>
   <Canvas>
      <Canvas.Resources>
         <ResourceDictionary>
            <SolidColorBrush x:Key="FavoriteColorFill"
               Color="#000000" />
         </ResourceDictionary>
      </Canvas.Resources>
      <!-- The following path is filed with color #000000 -->
      <Path Fill="{StaticResource FavoriteColorFill}">
         <Path.Data>
            ...
         </Path.Data>
      </Path>
      <Canvas>
         <!-- The following path is filed with color #000000 -->
         <Path Fill="{StaticResource FavoriteColorFill}">
            <Path.Data>
               ...
            </Path.Data>
         </Path>
      </Canvas>
   </Canvas>
   <!-- The following path is filled with color #808080 -->
   <Path Fill="{StaticResource FavoriteColorFill}">
      <Path.Data>
         ...
      </Path.Data>
   </Path>
</FixedPage>
```

end example]

14.2.6 Support for Markup Compatibility

If a resource dictionary contains Markup Compatibility elements and attributes, the processing of the Markup Compatibility markup MUST occur in the context of the definition of the resource dictionary, not in the context of resource references [M2.10].

14.3 Clipping

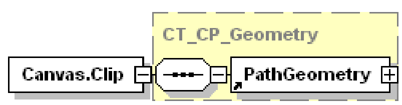
The Clip property specifies a geometric area that restricts the area to be filled with a brush.

The geometry is specified by a child <PathGeometry> element as detailed in §11.2, "Geometries and Figures," on page 77, or by abbreviated geometry syntax, described in §11.2.3, "Abbreviated Geometry Syntax," on page 90.

The default fill rule for geometries that do not specify a value is EvenOdd.

14.3.1 <Canvas.Clip> Element

element **Canvas.Clip**

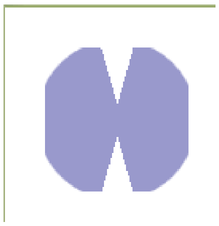
diagram	
annotation	Limits the rendered region of the element.

The <Canvas.Clip> property element applies to all child and descendant elements of the canvas.

Example 14–7. Canvas clip markup and rendering

```
<Canvas>
   <Canvas.Clip>
      <PathGeometry>
         <PathFigure StartPoint="25,25" IsClosed="true">
            <PolyLineSegment Points="60,25 70,60 80,25 115,25
               115,115 80,115 70,80 60,115 25,115" />
         </PathFigure>
      </PathGeometry>
   </Canvas.Clip>
   <Path Fill="#9999CC">
      <Path.Data>
         <PathGeometry>
            <PathFigure StartPoint="20,70">
               <ArcSegment
                  Point="120,70"
                  Size="50,50"
                  RotationAngle="0"
                  IsLargeArc="true"
                  SweepDirection="Clockwise" />
               <ArcSegment
                  Point="20,70"
                  Size="50,50"
                  RotationAngle="0"
                  IsLargeArc="true"
                  SweepDirection="Clockwise" />
            </PathFigure>
         </PathGeometry>
      </Path.Data>
   </Path>
</Canvas>
```

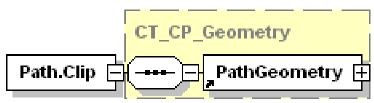
This markup is rendered as follows:



end example]

14.3.2 <Path.Clip> Element

element **Path.Clip**

diagram	
annotation	Limits the rendered region of the element.

A clipping region may also be applied to a specific path.

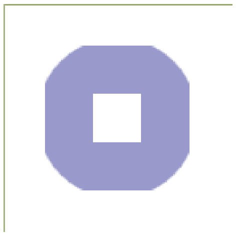
Example 14–8. <Path.Clip> usage

The following markup describes a complex clipping behavior:

```
<Path Fill="#9999CC">
   <Path.Clip>
      <PathGeometry>
         <PathFigure StartPoint="25,25" IsClosed="true">
            <PolyLineSegment Points="115,25 115,115 25,115" />
         </PathFigure>
         <PathFigure StartPoint="55,55" IsClosed="true">
            <PolyLineSegment Points="85,55 85,85 55,85" />
         </PathFigure>
      </PathGeometry>
   </Path.Clip>
   <Path.Data>
      <PathGeometry>
         <PathFigure StartPoint="20,70">
            <ArcSegment
               Point="120,70"
               Size="50,50"
               RotationAngle="0"
               IsLargeArc="true"
               SweepDirection="Clockwise" />
            <ArcSegment
               Point="20,70"
               Size="50,50"
               RotationAngle="0"
               IsLargeArc="true"
               SweepDirection="Clockwise" />
         </PathFigure>
      </PathGeometry>
   </Path.Data>
```

```
</Path>
```

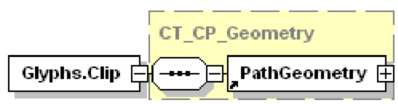
This markup is rendered as follows:



end example]

14.3.3 <Glyphs.Clip> Element

element **Glyphs.Clip**

diagram	
annotation	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.

Example 14–9. <Glyphs.Clip> usage

The following markup uses abbreviated geometry syntax to define the clipping region:

```
<Glyphs
   Fill="#000000"
   Clip="M 0,0 L 180,0 L 180,140 L 0,140 Z M 20,60 L 140,60 L 140,80
      L 20,80 Z"
   OriginX="20"
   OriginY="130"
   UnicodeString="N"
   FontRenderingEmSize="170"
   FontUri="../Resources/Fonts/Timesbd.ttf" />
```

This markup is rendered as follows:

end example]

14.4 Positioning Content

Content is positioned according to the properties specified for the fixed page or canvas, the properties specified for elements within the fixed page or canvas, and the compositional rules defined for the fixed payload namespace.

Elements are positioned relative to the current origin (0,0) of the coordinate space. The current origin can be moved by setting the RenderTransform property of a canvas, path, or glyph. The render transformation establishes a new coordinate frame for all children of the parent element.

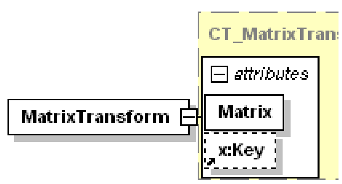
Geometries and brushes may be manipulated in a similar way by setting the Transform property. The transform results are concatenated with the current render transformation to create an effective render transformation for the local element.

The RenderTransform and Transform properties both specify an affine matrix transformation to the local coordinate space, using the <MatrixTransform> element as their value. An abbreviated matrix transformation syntax MAY be used to specify a **RenderTransform** or **Transform** attribute value [M2.72].

14.4.1 <MatrixTransform> Element

element **MatrixTransform**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
Matrix	ST_Matrix	required			Specifies the matrix structure that defines the transformation.
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 [M7.11].

annotation

Creates an arbitrary affine matrix transformation that manipulates objects or coordinate systems in a two-dimensional plane.

The <MatrixTransform> element defines an arbitrary affine matrix transformation used to manipulate the coordinate systems of elements. A 3x3 matrix is used for transformations in an x,y plane. Affine transformation matrices can be multiplied to form any number of linear transformations, such as rotation and skew (shear), followed by translation. An affine transformation matrix has its final column equal to 0,0,1, so only the members in the first two columns are specified.

$$\begin{bmatrix} \text{M11} & \text{M12} & \mathbf{0} \\ \text{M21} & \text{M22} & \mathbf{0} \\ \text{OffsetX} & \text{OffsetY} & \mathbf{1} \end{bmatrix}$$

This structure is specified by the **Matrix** attribute of the <MatrixTransform> element as the six numbers in the first two columns. [*Example*: "M11,M12,M21,M22,OffsetX,OffsetY". *end example*]

A matrix transform may also be specified as a RenderTransform or Transform property attribute using the following abbreviated matrix transformation syntax:

```
M11,M12,M21,M22,OffsetX,OffsetY
```

The values M11, M12, M21, and M22 control linear transformations such as rotation and skew, while OffsetX and OffsetY provide positional translation. Some typical affine matrix transformation examples follow.

Example 14–10. Matrix scaling

$$\begin{bmatrix} \textbf{X scale-factor} & 0 & 0 \\ 0 & \textbf{Y scale-factor} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

end example]

Example 14–11. Matrix reversing the x axis

$$\begin{bmatrix} \mathbf{-1} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

end example]

Example 14–12. Matrix reversing the y axis

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathbf{-1} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

end example]

Example 14–13. Matrix skewing

$$\begin{bmatrix} 1 & \textbf{Y skew-factor} & 0 \\ \textbf{X skew-} & 1 & 0 \end{bmatrix}$$

factor

0 0 1

end example]

Example 14–14. Matrix Rotating

$$\begin{bmatrix} \mathbf{cos\ \Theta} & \mathbf{sin\ \Theta} & 0 \\ \mathbf{-sin\ \Theta} & \mathbf{cos\ \Theta} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

end example]

Example 14–15. Matrix positioning

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \mathbf{OffsetX} & \mathbf{OffsetY} & 1 \end{bmatrix}$$

end example]

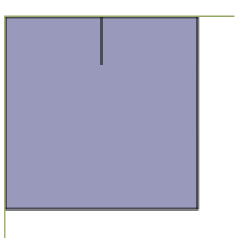
Example 14–16. <MatrixTransform> usage

The following markup describes a box (with the top edge marked) that is rotated 90° and shifted 50 units down and to the right:

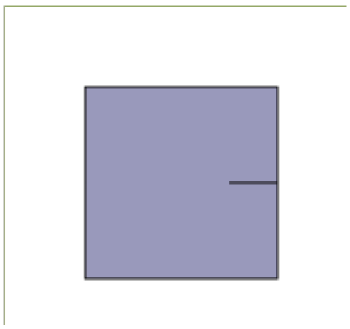
```
<Path
   Stroke="#000000"
   Fill="#9999BB"
   Data="M 0,0 L 60,0 L 60,25 L 60,0 L 120,0 L 120,120 L 0,120 Z">
   <Path.RenderTransform>
      <MatrixTransform Matrix="0,1,-1,0,170,50" />
   </Path.RenderTransform>
</Path>
```

Since the *x* origin has been shifted, the overall box must be additionally shifted the width of the box to achieve the desired visual effect.

Before the render transformation, the box appears like this:



After the render transformation, the box appears like this:



end example]

Example 14–17. Using abbreviated matrix transformation syntax

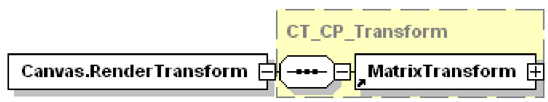
The following markup uses abbreviated syntax to produce the above image:

```
<Path
   Stroke="#000000"
   Fill="#9999BB"
   Data="M 0,0 L 60,0 L 60,25 L 60,0 L 120,0 L 120,120 L 0,120 Z"
   RenderTransform="0,1,-1,0,170,50" />
```

end example]

14.4.2 <Canvas.RenderTransform> Element

element **Canvas.RenderTransform**

diagram	
annotation	Establishes a new coordinate frame for the child and descendant elements of the canvas, such as another canvas. Also affects clip and opacity mask.

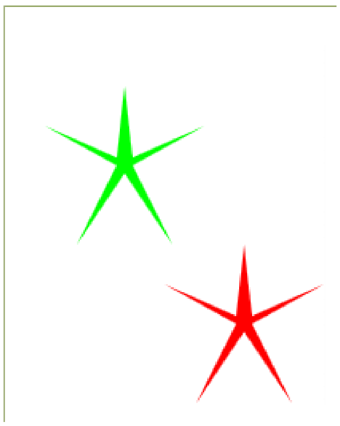
Example 14–18. <Canvas.RenderTransform> usage

In the following markup, child elements of the canvas are positioned by the render transformation:

```
<Canvas>
   <Canvas.Resources>
      <ResourceDictionary>
         <PathGeometry x:Key="StarFish">
            <PathFigure StartPoint="50,0" IsClosed="true">
               <PolyLineSegment Points="55,45 100,25 55,50 80,100 50,55
                  20,100 45,50 0,25 45,45" />
            </PathFigure>
         </PathGeometry>
      </ResourceDictionary>
   </Canvas.Resources>

   <!-- Draw a green starfish shifted 25 to the right and 50 down -->
   <Canvas>
      <Canvas.RenderTransform>
         <MatrixTransform Matrix="1,0,0,1,25,50" />
      </Canvas.RenderTransform>
      <Path Data="{StaticResource StarFish}">
         <Path.Fill>
            <SolidColorBrush Color="#00FF00" />
         </Path.Fill>
      </Path>
   </Canvas>

   <!-- Draw a red starfish shifted 100 to the right and 150 down -->
   <Canvas>
      <Canvas.RenderTransform>
         <MatrixTransform Matrix="1,0,0,1,100,150" />
      </Canvas.RenderTransform>
      <Path Data="{StaticResource StarFish}">
         <Path.Fill>
```

```
            <SolidColorBrush Color="#FF0000" />
         </Path.Fill>
      </Path>
   </Canvas>
</Canvas>
```

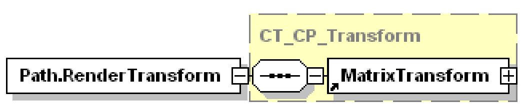
This markup is rendered as follows:



end example]

14.4.3 <Path.RenderTransform> Element

element **Path.RenderTransform**

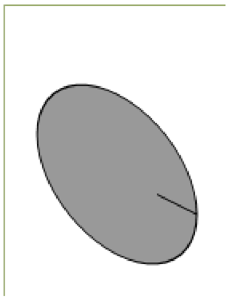
diagram	
annotation	Establishes a new coordinate frame for all attributes of the path and for all child elements of the path, such as the geometry defined by the <Path.Data> property element.

Example 14–19. <Path.RenderTransform> usage

The following markup describes a *y*-skew transformation applied to a circular path. (Before the render transformation, the middle of the right edge of the circle was marked with a horizontal line.)

```
<Path
   Fill="#999999"
   Stroke="#000000"
   Data="M 20,70 A 50,50 0 1 1 120,70 L 100,70 L 120,70 A 50,50 0 1 1
      20,70 Z"
   RenderTransform="1,0.5,0,1,0,0" />
```

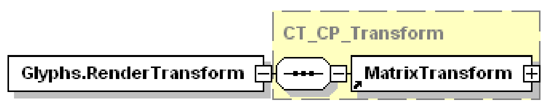
This markup is rendered as follows:



end example]

14.4.4 <Glyphs.RenderTransform> Element

element **Glyphs.RenderTransform**

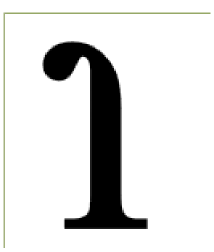
diagram	
annotation	Establishes a new coordinate frame for the glyph run specified by the <Glyphs> element. The render transform affects clip, opacity mask, fill, x origin, y origin, the actual shape of individual glyphs, and the advance widths. The render transform also affects the font size and values specified in the Indices attribute.

Example 14–20. <Glyphs.RenderTransform> usage

The following markup describes the letter J, flipped vertically and repositioned.

```
<Glyphs
   Fill="#000000"
   OriginX="20"
   OriginY="130"
   UnicodeString="J"
   FontRenderingEmSize="170"
   FontUri="../Resources/Fonts/Timesbd.ttf"
   RenderTransform="1,0,0,-1,0,150" />
```

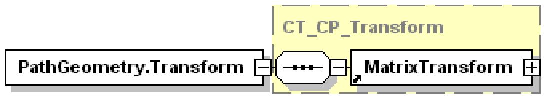
This markup is rendered as follows:



end example]

14.4.5 <PathGeometry.Transform> Element

element **PathGeometry.Transform**

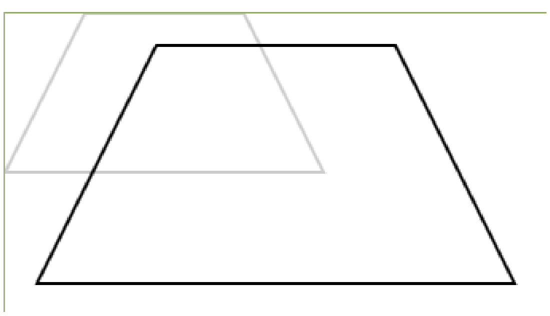
diagram	
annotation	Specifies the local matrix transformation that is applied to all child and descendant elements of the path geometry before it is used for filling, clipping, or stroking.

Example 14–21. <PathGeometry.Transform> usage

The following markup demonstrates a simple 150% zoom and positional transformation:

```
<Path StrokeThickness="2" Stroke="#000000">
   <Path.Data>
      <PathGeometry Transform="1.5,0,0,1.5,20,20">
         <PathFigure StartPoint="50,0" IsClosed="true">
            <PolyLineSegment Points="150,0 200,100 0,100" />
         </PathFigure>
      </PathGeometry>
   </Path.Data>
</Path>
```

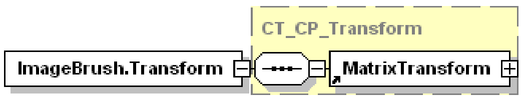
This markup is rendered as follows. The pre-transform path is indicated in light gray. Note that the stroke thickness did not change. If this transformation had been applied to the entire Path, the stroke thickness would also have increased by 150%.



end example]

14.4.6 <ImageBrush.Transform> Element

element **ImageBrush.Transform**

diagram	
annotation	Describes the matrix transformation applied to the coordinate space of the brush. The Transform property is concatenated with the current effective render transform to yield an effective render transform local to the brush. The viewport for the brush is transformed using the local effective render transform.

The **Transform** property can result in a non-rectangular (that is, skewed) viewport that defines the tile shape. In this circumstance, tile mode operations (FlipX, FlipY, and FlipXY) are treated as if the tile was rectangular, a larger tile was constructed from a 2-by-2 arrangement of regular tiles, the skew transform was applied afterward, and the new non-rectangular tile was tiled with adjacent edges and without flipping.

Example 14–22. <ImageBrush.Transform> usage

The following markup describes an image rotated 20° and repositioned within a path. The path itself remains untransformed; the viewport of the image brush is transformed instead.

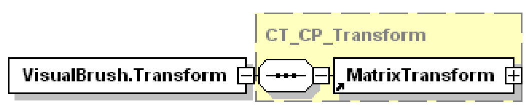
```
<Path
   StrokeThickness="5"
   Stroke="#996666"
   StrokeLineJoin="Round"
   Data="M 25,25 L 350,25 L 355,250 L 25,250 Z">
   <Path.Fill>
      <ImageBrush
         ImageSource="dog.jpg"
         Transform=".939,.342,-.342,.939,0,-80"
         TileMode="Tile"
         Viewbox="0,0,270,423"
         ViewboxUnits="Absolute"
         Viewport="75,75,90,125"
         ViewportUnits="Absolute" />
   </Path.Fill>
</Path>
```

This markup is rendered as follows:

end example]

14.4.7 <VisualBrush.Transform> Element

element **VisualBrush.Transform**

diagram	
annotation	Describes the matrix transformation applied to the coordinate space of the brush. The Transform property is concatenated with the current effective render transform to yield an effective render transform local to the brush. The viewport for the brush is transformed using the local effective render transform.

The **Transform** property can result in a non-rectangular (that is, skewed) viewport that defines the tile shape. In this circumstance, tile mode operations (FlipX, FlipY, and FlipXY) are treated as if the tile was rectangular, a larger tile was constructed from a 2-by-2 arrangement of regular tiles, the skew transform was applied afterward, and the new non-rectangular tile was tiled with adjacent edges and without flipping.

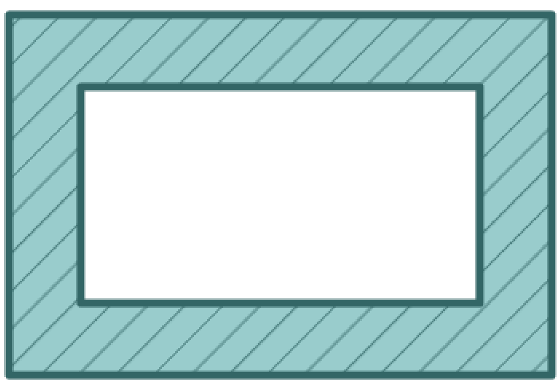
Example 14–23. <VisualBrush.Transform> usage

The following markup describes a solid background and vertical pinstripe rotated 45° to fill a frame:

```
<Path
   StrokeThickness="5"
   Stroke="#336666"
   StrokeLineJoin="Round"
   Data="M 25,25 L 365,25 L 365,250 L 25,250 Z M 70,70 L 320,70
      L 320,205 L 70,205 Z">
   <Path.Fill>
      <VisualBrush
```

```
         Transform=".707,.707,-.707,.707,0,0"
         TileMode="Tile"
         Viewbox="0,0,60,100"
         ViewboxUnits="Absolute"
         Viewport="25,25,50,50"
         ViewportUnits="Absolute">
         <VisualBrush.Visual>
            <Canvas>
               <Path
                  Fill="#99CCCC"
                Data="M 0,0 L 60,0 L 60,100 L 0,100 Z" />
               <Path
                  Stroke="#336666"
                  Data="M 0,0 L 0,100 M 20,0 L 20,100 M 40,0 L 40,100
                     M 60,0 L 60,100 M 80,0 L 80,100" />
            </Canvas>
         </VisualBrush.Visual>
      </VisualBrush>
   </Path.Fill>
</Path>
```

This markup is rendered as follows:



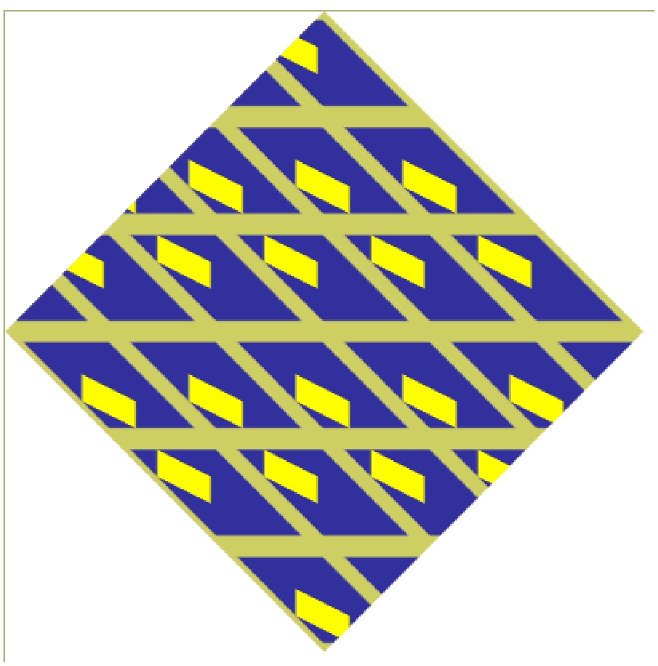
end example]

Example 14–24. <VisualBrush.Transform> usage with tiling behavior

This example demonstrates tile rendering behavior when applying a transform.

```
<!-- Draw background diamond to show where fill affects background -->
<Path Fill="#CCCC66" Data="M 200,0 L 400,200 L 200,400 L 0,200 Z" />
<Path Data="M 200,0 L 400,200 L 200,400 L 0,200 Z">
   <Path.Fill>
      <VisualBrush
         Viewbox="0,0,1,1"
         Viewport="200,133,67,67"
         ViewboxUnits="Absolute"
         ViewportUnits="Absolute"
         Transform="1,0,1,1,0,0"
         TileMode="FlipY">
         <VisualBrush.Visual>
            <Canvas>
               <Path Fill="#333399" Data="M 0.1,0.1 L 0.9,0.1 L 0.9,0.9
                  L 0.1,0.9 Z" />
               <Path Fill="#FFFF00" Data="M 0.1,0.35 L 0.35,0.1
                  L 0.6,0.35 L 0.35,0.6 Z" />
            </Canvas>
         </VisualBrush.Visual>
      </VisualBrush>
   </Path.Fill>
</Path>
```

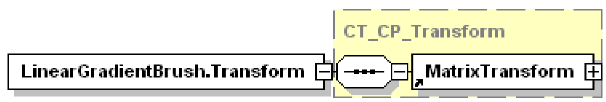
This markup is rendered as follows:



end example]

14.4.8 <LinearGradientBrush.Transform> Element

element **LinearGradientBrush.Transform**

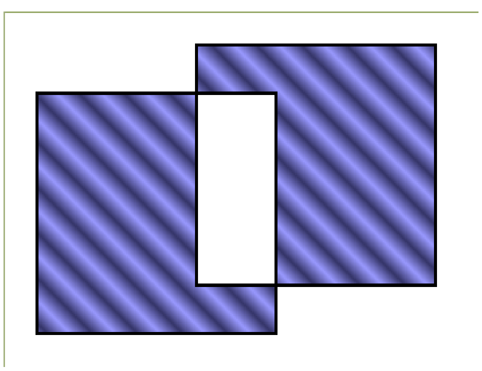
diagram	
annotation	Describes the matrix transformation applied to the coordinate space of the brush. The Transform property is concatenated with the current effective render transform to yield an effective render transform local to the brush. The start point and end point are transformed using the local effective render transform.

Example 14–25. <LinearGradientBrush.Transform> usage

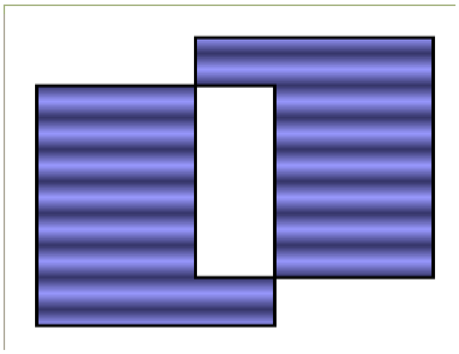
The following markup demonstrates a transform applied to the brush directly:

```
<Path Stroke="#000000" StrokeThickness="2" Data="M 20,50 L 170,50 L
170,200 L 20,200 Z M 120,20 L 270,20 L 270,170 120,170 Z">
   <Path.Fill>
      <LinearGradientBrush
         MappingMode="Absolute"
         Transform=".707,.707,-.707,.707,150,-30"
         StartPoint="0,0"
         EndPoint="0,10"
         SpreadMethod="Reflect">
         <LinearGradientBrush.GradientStops>
            <GradientStop Color="#9999FF" Offset="0.0"/>
            <GradientStop Color="#333366" Offset="1.0"/>
         </LinearGradientBrush.GradientStops>
      </LinearGradientBrush>
   </Path.Fill>
</Path>
```

This markup is rendered as follows:



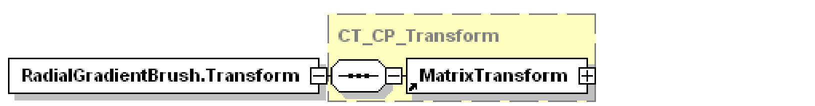
Without the Transform property, this markup would be rendered as follows:



end example]

14.4.9 <RadialGradientBrush.Transform> Element

element **RadialGradientBrush.Transform**

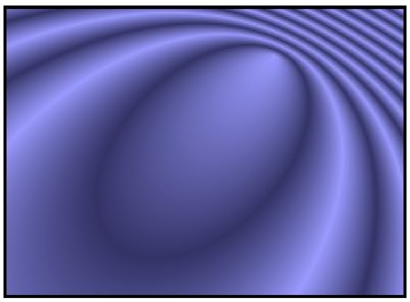
diagram	
annotation	Describes the matrix transformation applied to the coordinate space of the brush. The Transform property is concatenated with the current effective render transform to yield an effective render transform local to the brush. The ellipse defined by the center, gradient origin, x radius, and y radius vaules is transformed using the local effective render transform.

Example 14–26. <RadialGradientBrush.Transform> usage

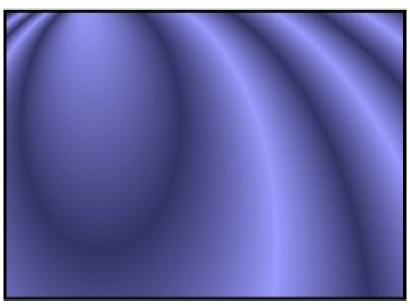
The following markup describes a rotation and reposition transform on a radial gradient:

```
<Path
   Stroke="#000000"
   StrokeThickness="2"
   Data="M 20,20 L 270,20 L 270,200 L 20,200 Z">
   <Path.Fill>
      <RadialGradientBrush
         MappingMode="Absolute"
         Transform=".707,.707,-.707,.707,150,-10"
         Center="80,90"
         RadiusX="50"
         RadiusY="80"
         GradientOrigin="70,15"
         SpreadMethod="Reflect">
         <RadialGradientBrush.GradientStops>
            <GradientStop Color="#9999FF" Offset="0.0" />
            <GradientStop Color="#333366" Offset="1.0" />
         </RadialGradientBrush.GradientStops>
      </RadialGradientBrush>
   </Path.Fill>
</Path>
```

This markup is rendered as follows:



Without the Transform property, this markup is rendered as follows:



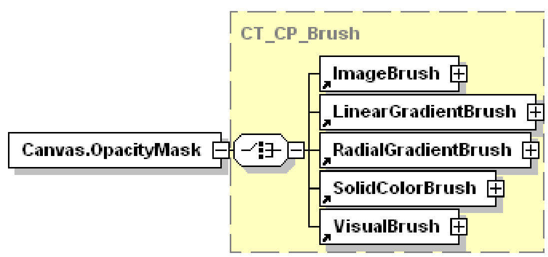
end example]

14.5 OpacityMask

The OpacityMask property defines a variable alpha mask for the parent element. The alpha for areas not marked by the brush is 0.0.

14.5.1 <Canvas.OpacityMask> Element

element **Canvas.OpacityMask**

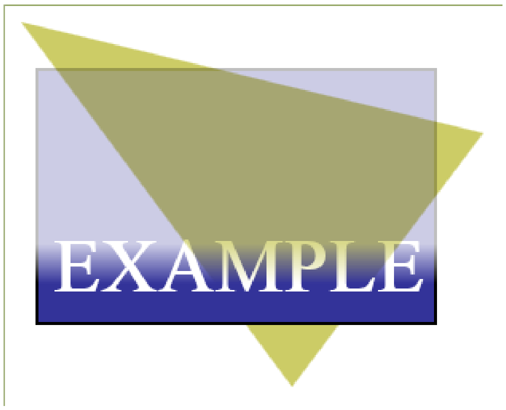
diagram	
annotation	Specifies a mask of alpha values that is applied to the canvas in the same fashion as the Opacity attribute, but allowing different alpha values for different areas of the element.

Example 14–27. <Canvas.OpacityMask> usage

In the following markup, the contents of the canvas are opaque with respect to each other, but both elements are blended with the background triangle:

```
<Path Fill="#CCCC66" Data="M 10,10 L 300,80 L 180,240 Z" />
<Canvas>
   <Canvas.OpacityMask>
      <LinearGradientBrush
         MappingMode="Absolute"
         StartPoint="0,150"
         EndPoint="0,175"
         SpreadMethod="Pad">
         <LinearGradientBrush.GradientStops>
            <GradientStop Color="#40000000" Offset="0.0" />
            <GradientStop Color="#FF000000" Offset="1.0" />
         </LinearGradientBrush.GradientStops>
      </LinearGradientBrush>
   </Canvas.OpacityMask>
   <Path
      Stroke="#000000"
      StrokeThickness="2"
      Fill="#333399"
      Data="M 20,40 L 270,40 L 270,200 L 20,200 Z" />
   <Glyphs
      OriginX="30"
      OriginY="180"
      UnicodeString="EXAMPLE"
      FontUri="../Resources/Fonts/Timesbd.ttf"
      FontRenderingEmSize="48"
      Fill="#FFFFFF" />
</Canvas>
```

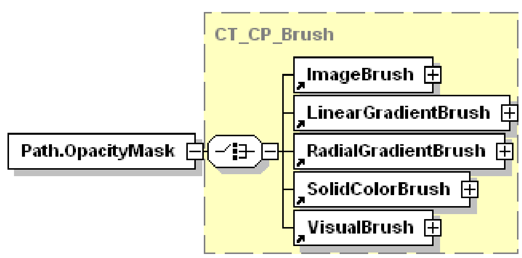
This markup is rendered as follows:



end example]

14.5.2 <Path.OpacityMask> Element

element **Path.OpacityMask**

diagram	
annotation	Specifies the mask of alpha values that is applied to the path in the same fashion as the Opacity attribute, but allowing different alpha values for different areas of the element.

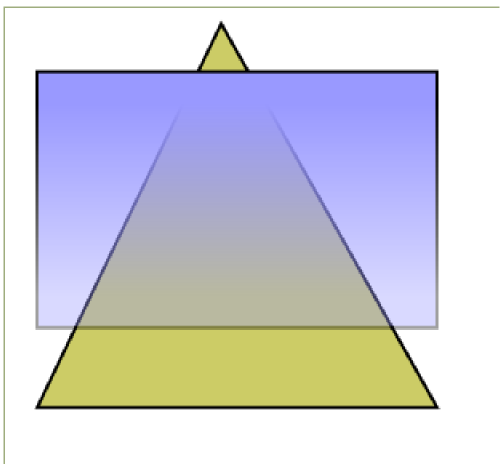
Example 14–28. <Path.OpacityMask> usage

The following markup describes a path that has a linear gradient both for the opacity mask and for the fill:

```
<Path
   Stroke="#000000"
   StrokeThickness="2"
   Fill="#CCCC66"
   Data="M 135,10 L 270,250 L 20,250 Z" />
<Path
   Stroke="#000000"
   StrokeThickness="2"
   Data="M 20,40 L 270,40 L 270,200 L 20,200 Z">
   <Path.OpacityMask>
      <LinearGradientBrush
         MappingMode="Absolute"
         StartPoint="0,60"
```

```
         EndPoint="0,180"
         SpreadMethod="Pad">
         <LinearGradientBrush.GradientStops>
            <GradientStop Color="#FF000000" Offset="0.0" />
            <GradientStop Color="#60000000" Offset="1.0" />
         </LinearGradientBrush.GradientStops>
      </LinearGradientBrush>
   </Path.OpacityMask>
   <Path.Fill>
      <SolidColorBrush Color="#9999FF" />
   </Path.Fill>
</Path>
```

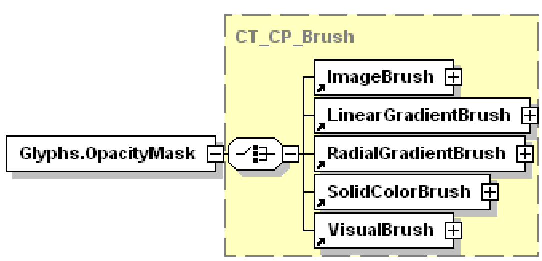
This markup is rendered as follows:



end example]

14.5.3 <Glyphs.OpacityMask> Element

element **Glyphs.OpacityMask**

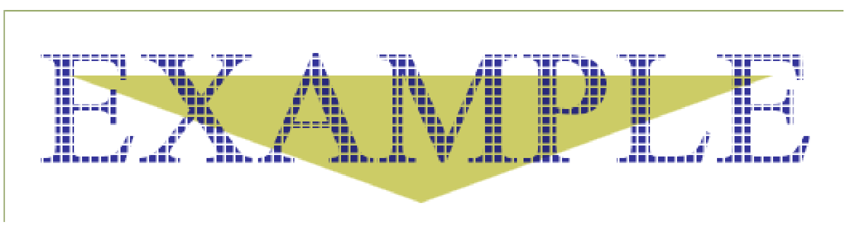
diagram	
annotation	Specifies a mask of alpha values that is applied to the glyphs in the same fashion as the Opacity attribute, but allowing different alpha values for different areas of the element.

Example 14–29. <Glyphs.OpacityMask> usage

The following markup demonstrates the use of an opacity mask to create a tile effect:

```
<Path Fill="#CCCC66" Data="M 40,40 L 480,40 L 260,120 Z" />
<Glyphs
   OriginX="20"
   OriginY="95"
   UnicodeString="EXAMPLE"
   FontUri="../Resources/Fonts/Timesbd.ttf"
   FontRenderingEmSize="100"
   Fill="#000080">
   <Glyphs.OpacityMask>
      <VisualBrush
         Viewbox="0,0,2,2"
         ViewboxUnits="Absolute"
         Viewport="0,0,6,6"
         ViewportUnits="Absolute"
         TileMode="Tile">
         <VisualBrush.Visual>
            <Path
               Fill="#CC000000"
               Data="M 0,0 L 1.5,0 L 1.5,1.5 L 0,1.5 Z" />
         </VisualBrush.Visual>
      </VisualBrush>
   </Glyphs.OpacityMask>
</Glyphs>
```

This markup is rendered as follows:



end example]

15. Color

The mechanisms described in this clause for storing advanced color information in XPS Documents apply to both vector graphics (including text) and raster images. Color producers such as digital cameras and consumers such as printers can store and render significantly more color information than many display devices can render (typically 8 bits per channel). Storing the advanced color information in an XPS Document and passing it through to printing consumers enables greater end-to-end color fidelity.

15.1 Color Support

XPS Documents support sRGB and rich color spaces, including scRGB, CMYK, N-Channel, and named colors. Consumers MUST support the following color features:

- sRGB colors (8 bit-per-channel) in vector data, with and without alpha [M8.1]
- sRGB colors in image data, using the JPEG, PNG, TIFF, or Windows Media Photo image formats [M8.2]
- scRGB color specification in vector data, with and without alpha [M8.3]
- scRGB colors in image data, using the Windows Media Photo image format [M8.4]
- CMYK colors in vector data [M8.5]
- CMYK colors in image data, using the TIFF or Windows Media Photo image formats [M8.6]
- N-Channel colors in vector data [M8.7]
- N-Channel colors in image data, using the Windows Media Photo image format [M8.8]
- ICC Version 2 profiles for 3-, 4-, 5-, 6-, 7-, and 8-channel color [M8.9]
- ICC Version 2 profiles with a Windows Color System (WCS) profile embedded as a private tag [M8.10]

When non-sRGB color information is used, color value specifications are expressed using markup from the XPS Document schema.

Consumers are not required to handle all color spaces natively, but rather MAY convert data specified in a rich color space to sRGB at an early stage [O8.1]. Consumers that do not handle rich colors natively may experience reduced fidelity.

15.1.1 sRGB Color Space

The XPS Document format supports colors in the sRGB color space for both vector and raster graphics.

15.1.2 scRGB Color Space

The XPS Document format supports colors in the scRGB color space for both vector and raster graphics. See G for the scRGB gamut definition.

15.1.3 Gray Color Space

Gray colors for vector elements can be specified as sRGB or scRGB colors with the red, blue and green components set to the same value. Gray colors for raster images can be specified using any image format.

15.1.4 CMYK Color Space

CMYK color is supported through the use of color management transformations from an ICC profile.

15.1.5 N-Channel Color Spaces

N-channel color is supported through the use of color management transformations from an ICC profile.

15.1.6 Named Color for Spot Colors and N-tone Images

Named colors are supported for images and spot colors through the use of color management transformations from an ICC profile.

15.1.7 Device Color Spaces

To specify colors in the native color space (usually CMYK or N-Channel) for a device, use the standard markup with an ICC profile that approximates the device color space. Include the PageDeviceColorSpaceProfileURI PrintTicket setting to indicate to the device that this particular color specification is a native device color and MUST NOT be color-managed according to the included profile unless forced to do so for transparency effects [M8.11].

It is the responsibility of the consumer to identify the ICC profile as one that correctly approximates the native colors of the device.

15.1.8 ICC Profiles

XPS Documents MAY include ICC profile parts [O2.3]. ICC profiles MUST conform to the requirements in the ICC Color Profile specification, Version 3.4 [M8.12].

Consumers MUST support color profiles as specified in the ICC specification [M8.9]. However, the set of usable N-component LUT-based profiles is limited to 3-, 4-, 5-, 6-, 7-, or 8-color channels.

All ICC profiles used in XPS Documents MUST be one of the following [M8.13]:

- Input
- Output
- Monitor (RGB)
- ColorSpace Conversion

One tag is explicitly supported as specified in ICC Version 4.0.0 profiles: the colorant table for named colors. A consumer incapable of supporting named colors SHOULD treat this tag as a user-defined custom tag, and therefore ignore it. It SHOULD instead use the color tables as provided in the profile to convert the specified colors to the Profile Connection Space (PCS) [S8.14].

ICC profiles SHOULD be used when embedded in any image format with any color space, except the scRGB color space, for which the gamut boundary described in G is assumed [S8.1].

[*Note*: Some consumers do not correctly apply ICC profiles to grayscale images. *end note*] If consistency of appearance is important, the producer SHOULD adjust the gray tone response curve of the image before adding it to the XPS Document [S8.2].

An ICC profile MAY contain the private tag, “MS00”, which specifies an embedded Windows Color System (WCS) profile [O8.2].

15.1.9 WcsProfilesTag

The WcsProfilesTag (Windows color space profiles tag) is a private Microsoft ICC profile tag that is used in ICC profiles created by WcsCreateIccProfile to contain input WCS profiles. This tag conforms to ICC profile requirements for profile tags. In particular, the tag header is in big-endian byte ordering, but the embedded WCS XML profiles remain in their native byte order. Furthermore, the tag data must be aligned on a 4-byte boundary (measured from the start of the ICC profile). The structure of the tag is defined by the WcsProfilesTagType below.

The WcsProfilesTag signature is “MS00”. This is the tag signature that will appear in the ICC profiles tag table for the WcsProfilesTag.

Table 15–3. WcsProfilesTagType structure

Byte offset	Content
0–3	MS10-type signature.
4–7	Reserved, must be set to 0 (according to ICC convention).
8–11	Byte offset from the beginning of the tag to the CDMP data.
12–15	Size of the CDMP data in bytes.
16–19	Byte offset from the beginning of the tag to the CAMP data.
20–23	Size of the CAMP data in bytes.
24–27	Byte offset from the beginning of the tag to the GMMP data.
28–31	Size of the GMMP data in bytes.
32–*n*	A sequence of bytes of length `element_size-32`, where `element_size` is the tag size recorded in the ICC profile tag table entry for this tag. These are the WCS XML profiles that were used to create this ICC profile via WcsCreateIccProfileFromWcsProfileHandle. The WCS profiles are ordered as follows: the CDMP (required) first, followed by the CAMP (if present), followed by the GMMP (if present).

15.1.10 WCS Color Profiles

XPS Documents include only WCS color profiles embedded in ICC color profiles, as described above.

15.1.11 Vector Color Syntax

Vector colors may be specified in XPS Document markup in the following locations:

- The **Color** attribute of the <SolidColorBrush> element
- The **Color** attribute of the <GradientStop> element
- The **Fill** attribute of the <Path> element
- The **Fill** attribute of the <Glyphs> element
- The **Stroke** attribute of the <Path> element

The last three locations are an abbreviated syntax for expressing a solid color brush with the specified color.

Table 15–1. Syntax summary

Color type	Syntax	Example
sRGB w/o alpha	`Color="#RRGGBB"`	`Color="#FFFFFF"`
sRGB with alpha	`Color="#AARRGGBB"`	`Color="#80FFFFFF"`
scRGB w/o alpha	`Color="sc#RedFloat, GreenFloat,BlueFloat"`	`Color="sc#1.0,0.5,1.0"`
scRGB with alpha	`Color="sc#AlphaFloat,RedFloat, GreenFloat,BlueFloat"`	`Color="sc#0.3,1.0,0.5,1.0"`
CMYK with alpha	`Color="ContextColor ProfileURI AlphaFloat, Chan0Float, Chan1Float, Chan2Float, Chan3Float"`	`Color="ContextColor /swopcmykprofile.icc 1.0,1.0,0.0,0.0,0.0"`
N-Channel with alpha	`Color="ContextColor ProfileURI AlphaFloat, Chan0Float, ..., ChanN-1Float"`	`Color="ContextColor /5nchannelprofile.icc 1.0, 1.0, 0.0, 0.0, 1.0, 0.0"`
Named color with alpha	`Color="ContextColor ProfileURI AlphaFloat, TintFloat, 0, 0"`	`Color="ContextColor /namedtintprofile.icc 1.0, 1.0, 0, 0"`

Real numbers specified for color channel values of scRGB and ContextColor colors MUST NOT use exponent forms of numbers [M8.14].

15.1.12 sRGB Color Syntax

The sRGB color syntax is the same as that used in HTML, with the red, green, and blue channels represented by two hexadecimal digits. XPS Documents may specify an sRGB color either with or without an alpha channel value, which is also expressed as two hexadecimal digits.

The syntax is as follows (without alpha):

```
#RRGGBB
```

or (with alpha):

```
#AARRGGBB
```

When an sRGB color is specified without an alpha value, an alpha of “FF” is implied.

15.1.13 scRGB Color Syntax

The scRGB color syntax allows XPS Document producers to specify a color using the full scRGB color space, which is much larger than the sRGB color space and can represent the entire range of colors perceivable by the human eye.

This syntax is expressed either as:

```
sc#RedFloat,GreenFloat,BlueFloat
```

or:

```
sc#AlphaFloat,RedFloat,GreenFloat,BlueFloat
```

When an scRGB color is specified with three numeric values, an alpha of 1.0 is implied. When an scRGB color is specified with four numeric values, the first value is the alpha channel. Although alpha values smaller than 0.0 and larger than 1.0 can be specified, they MUST be clamped to the valid range from 0.0 to 1.0 before any further processing [M8.15].

15.1.14 CMYK Color Syntax

XPS Document producers specify CMYK colors using the context color syntax, which allows specification of an ICC profile and the individual color channel values as real numbers.

The syntax is as follows:

```
ContextColor ProfileURI AlphaFloat, Chan0Float, Chan1Float, Chan2Float,
Chan3Float
```

`ProfileURI` specifies a part containing the binary data of the color profile. The profile URI MUST be added as a Required Resource relationship to the FixedPage part [M2.10].

Although alpha values smaller than 0.0 and larger than 1.0 can be specified, they MUST be clamped to the valid range from 0.0 to 1.0 before any further processing [M8.16]. Channel float values MUST also be clamped to the valid range from 0.0 to 1.0 before further processing. If the value is used as input for an ICC profile color transformation, it MUST subsequently be linearly scaled to the range from 0 to 255 or from 0 to 65535, depending on whether the profile uses 8-bit or 16-bit input tables [M8.31].

15.1.15 N-Channel Color Syntax

XPS Document producers specify N-channel colors using the context color syntax, which allows specification of an ICC profile and the individual color channel values as real numbers. The syntax is expressed as follows:

```
ContextColor ProfileURI AlphaFloat,Chan0Float,...,ChanN-1Float
```

`ProfileURI` specifies a part containing the binary data of the color profile. The profile URI MUST be added as a Required Resource relationship to the FixedPage part [M2.10]. The profile can be a 3-, 4-, 5-, 6-, 7- or 8-channel profile. The context color MUST specify a matching number of channel float values, setting unused ones to 0.0 [M8.17].

Although alpha values smaller than 0.0 and larger than 1.0 can be specified, they MUST be clamped to the valid range from 0.0 to 1.0 before any further processing [M8.18]. Channel float values MUST also be clamped to the valid range from 0.0 to 1.0 before further processing. If the value is used as input for an ICC profile color transformation, it MUST subsequently be linearly scaled to the range from 0 to 255 or from 0 to 65535, depending on whether the profile uses 8-bit or 16-bit input tables [M8.31].

To represent a 1-channel profile, create a 3-channel profile and use only the first channel. To represent a 2-channel profile, create a 3-channel profile and use only the first and second channels.

15.1.16 Named Color Syntax

A *named color* is an industry-defined color specification that identifies a particular color in a well-defined color schema, usually for the purpose of printing. There are currently several named color schemas. Producers specify named colors using the context color syntax, which allows specification of a named color using an ICC profile. The named color schema used is determined by the ICC profile.

A named color is expressed as a combination of an ink name stored in the ICC profile and a tint level (percentage ink dilution). The ink name for the tint is contained in the colorantTable clrt tag. This tag is not defined for ICC Version 3.4 profiles, and its presence is benignly ignored by ICM V2 implementations. Therefore, it is used in XPS Documents to specify the names of named colors.

The syntax for referencing a single named color is as follows:

```
ContextColor ProfileURI AlphaFloat,TintFloat,0,0
```

`ProfileURI` specifies a part containing the binary data of the color profile. The profile URI MUST be added as a Required Resource relationship to the FixedPage part [M2.10]. The profile MUST have 3, 4, 5, 6, 7 or 8 channels (and an *n*CLR signature, where *n* is the number of channels), mapping to a valid PCS [M8.19]. To represent a single tone or duotone profile, create a 3-channel profile and specify a colorant name in the profile's colorantTable with zero-length (name field all set to 0) for the unused channels.

`AlphaFloat` specifies the alpha to be applied to the named color. `TintFloat` specifies how diluted with respect to the color schema's white color point the named color is, with `1.0` being the pure named color and `0.0` being fully diluted. The final two values are always set to `0` to specify a single named color.

For duotone named colors, use the first two values and set the final value to `0`. [*Example*: The syntax for referencing a duotone named color is as follows:

```
ContextColor ProfileURI AlphaFloat,Tint0Float,Tint1Float,0
```

end example]

Although alpha values smaller than 0.0 and larger than 1.0 can be specified, they MUST be clamped to the valid range from 0.0 to 1.0 before any further processing [M8.20]. The tint float value MUST also be clamped to the valid range from 0.0 to 1.0 before further processing. If the value is used as input for an ICC profile color transformation, it MUST subsequently be linearly scaled to the range from 0 to 255 or from 0 to 65535, depending on whether the profile uses 8-bit or 16-bit input tables [M8.31].

Consumers that do not understand the named color MUST compute a color approximation through ICM-compliant color management functions using the specified profile. When a named color is used in a gradient brush or with transparency, the result produced by consumers that are not named-color aware MAY differ significantly from the result produced by consumers that are named-color aware [O8.3].

An XPS consumer that is aware of named colors looks for the clrt tag to find out if a specific ContextColor designates a named color description. A producer of XPS documents containing named colors SHOULD create the color profile in such a way that a linear ramp of the channel values corresponding to a named colorant maps to PCS values resulting in the same color appearance for consumers unaware of named colors (or the specific colorant) [S8.3].

This method can be used for 1 to 8 named colors; however, the ContextColor syntax requires a minimum of 1 Alpha value and 3 Channel values. It is RECOMMENDED that a 1 or 2 tone profile uses the first 1 or 2 channels, respectively, and specifies 0 for the remaining channels [S8.4].

If the consumer does not know ALL of the colorants named in the clrt tag, it SHOULD treat the profile as if it were a regular N-channel source profile and SHOULD NOT attempt to use any of the known colorants, as that would result in undefined results [S8.6].

A named color profile MAY be used with images [O8.4], which is especially useful for the reproduction of corporate logo images using strictly controlled ink sets. A named profile could contain SWOP CMYK in the first 4 channels, and use the remaining 4 channels for highlight colors. This allows the use of up to 4 accent colors in images.

15.2 Rich Colors in Raster Images

This section describes specific considerations for including raster images with rich colors in XPS Documents.

15.2.1 sRGB Raster Images

XPS Documents support sRGB raster images in the following formats:

- JPEG
- PNG
- TIFF
- Windows Media Photo

The following Windows Media Photo pixel formats are supported:

- WICPixelFormat24bppRGB
- WICPixelFormat24bppBGR
- WICPixelFormat32bppBGR
- WICPixelFormat32bppBGRA
- WICPixelFormat32bppPBGRA
- WICPixelFormat48bppRGB
- WICPixelFormat64bppRGBA
- WICPixelFormat64bppPRGBA

Pixel formats WICPixelFormat32bppPBGRA and WICPixelFormat64bppPRGBA are pre-multiplied alpha formats. See §18.4.1, “Pre-Multiplied Alpha and Superluminous Colors,” on page 281 for details.

The following Windows Media Photo packed pixel formats are supported:

- WICPixelFormat16bppBGR555
- WICPixelFormat16bppBGR565
- WICPixelFormat32bppBGR101010

See §9.1.5, "Image Parts," on page 24 for more details.

15.2.2 scRGB Raster Images

XPS Documents support scRGB raster images only in the Windows Media Photo image format. The following pixel formats are supported:

- WICPixelFormat48bppRGBFixedPoint
- WICPixelFormat48bppRGBHalf
- WICPixelFormat96bppRGBFixedPoint
- WICPixelFormat128bppRGBFloat
- WICPixelFormat64bppRGBAFixedPoint
- WICPixelFormat64bppRGBFixedPoint
- WICPixelFormat64bppRGBAHalf
- WICPixelFormat64bppRGBHalf
- WICPixelFormat128bppRGBAFixedPoint
- WICPixelFormat128bppRGBFixedPoint
- WICPixelFormat128bppRGBAFloat
- WICPixelFormat128bppPRGBAFloat
- WICPixelFormat32bppRGBE

Pixel format WICPixelFormat128bppPRGBAFloat is a pre-multiplied alpha format. See 18.4.1, "Pre-Multiplied Alpha and Superluminous Colors," on page 281 for details.

15.2.3 Gray Raster Images

XPS Documents support gray raster images in the following formats:

- JPEG
- PNG
- TIFF
- Windows Media Photo

The following Windows Media Photo pixel formats are supported:

- WICPixelFormatBlackWhite
- WICPixelFormat8bppGray
- WICPixelFormat16bppGray
- WICPixelFormat16bppGrayFixedPoint (scRGB range)
- WICPixelFormat16bppGrayHalf (scRGB range)

- WICPixelFormat32bppGrayFixedPoint (scRGB range)
- WICPixelFormat32bppGrayFloat

15.2.4 CMYK Raster Images

CMYK images are stored in TIFF 6.0 or Windows Media Photo format.

15.2.4.1 TIFF CMYK Raster Images

CMYK TIFF image tags are described in §9.1.5.3, "TIFF Images."

ICC profiles can be associated with CMYK raster images by using an ICC profile embedded in the TIFF file (tag 34675) or associated using the mechanism described in §15.2.8, "Images and Color Profile Association," on page 216.

15.2.4.2 Windows Media Photo CMYK Raster Images

The Windows Media Photo CMYK format is described in the Windows Media Photo specification. The following formats are supported:

- WICPixelFormat32bppCMYK
- WICPixelFormat40bppCMYKAlpha
- WICPixelFormat64bppCMYK
- WICPixelFormat80bppCMYKAlpha

15.2.4.3 JPEG CMYK Raster Images

Support for JPEG CMYK images varies by implementation and SHOULD NOT be used in XPS Documents [S2.7]. See §9.1.5.1, "JPEG Images" for more details.

15.2.5 N-channel Raster Images

N-channel images are stored in the Windows Media Photo image file format using an ICC profile. The following formats are supported:

- WICPixelFormat24bpp3Channels, WICPixelFormat48bpp3Channels
- WICPixelFormat32bpp4Channels, WICPixelFormat64bpp4Channels
- WICPixelFormat40bpp5Channels, WICPixelFormat80bpp5Channels
- WICPixelFormat48bpp6Channels, WICPixelFormat96bpp6Channels
- WICPixelFormat56bpp7Channels, WICPixelFormat112bpp7Channels
- WICPixelFormat64bpp8Channels, WICPixelFormat128bpp8Channels
- WICPixelFormat32bpp3ChannelsAlpha, WICPixelFormat64bpp3ChannelsAlpha
- WICPixelFormat40bpp4ChannelsAlpha, WICPixelFormat80bpp4ChannelsAlpha
- WICPixelFormat48bpp5ChannelsAlpha, WICPixelFormat96bpp5ChannelsAlpha
- WICPixelFormat56bpp6ChannelsAlpha, WICPixelFormat112bpp6ChannelsAlpha
- WICPixelFormat64bpp7ChannelsAlpha, WICPixelFormat128bpp7ChannelsAlpha
- WICPixelFormat72bpp8ChannelsAlpha, WICPixelFormat144bpp8ChannelsAlpha

15.2.6 Named Color Raster Images

Named color (N-tone) raster images are stored in the Windows Media Photo image file format using an ICC profile that maps the tint channel combinations to valid PCS values. See §15.2.5, “N-channel Raster Images,” on page 215 for pixel format definitions.

Consumers unaware of named colors can then compute color approximations using the PCS values computed from the profile.

15.2.7 Device Color Raster Images

Device color (N-channel) raster images are stored in the Windows Media Photo image file format in the same manner as a named color raster image. See §15.1.7, “Device Color Spaces,” on page 206 for more details. RGB and CMYK raster images can also be stored in the TIFF image file format. JPEG CMYK images SHOULD NOT be used [S2.7].

15.2.8 Images and Color Profile Association

Images can use a color profile matching the channel configuration of the image using one of two methods:

- Color profile embedded in an image using the image format specific mechanism
- Color profile contained in a separate part associated with the image using the following markup:

```
<ImageBrush ImageSource="{ColorConvertedBitmap image.tif profile.icc}"
... />
```

  An associated color profile overrides an embedded color profile and is processed instead of any embedded color profile. The profile URI MUST be added as a Required Resource relationship to the FixedPage part [M2.10].

15.2.9 Color Space Pixel Formats for Raster Images

If an ICC profile is not embedded or associated with a raster image or if the embedded or associated profile is not compatible with the pixel format of the image, the default pixel formats for each color space MUST be treated as follows [M8.30].

Table 15–2. Color Space Pixel Format Defaults

Color Space	Pixel Formats
sRGB	Integer RGB Integer Grayscale Integer 3-Channel
scRGB (wcsRGB gamut)	Floating Point scRGB Half-Float scRGB Fixed-Point scRGB Floating Point Grayscale Half-Float Grayscale Fixed-Point Grayscale

CMYK (with SWOP profile)	Integer CMYK Integer 4-Channel Integer 5-Channel (ignore channel 5) Integer 6-Channel (ignore channels 5 and 6) Integer 7-Channel (ignore channels 5, 6, and 7) Integer 8-Channel (ignore channels 5, 6, 7, and 8)

15.3 Color Separation

Consumers MAY perform color separation, if desired [O8.5].

A named color used for markings that are intended to be rendered on every layer of the separation can be specified with the DocumentImpositionColor PrintTicket setting.

The color name specified by the DocumentImpositionColor PrintTicket setting MUST be matched only to profiles containing exactly one non-zero-length colorant name in the profile's colorantTable [M8.22]. The color name specified by the DocumentImpositionColor setting serves as a label for that color only and MUST NOT be matched against any Named Colors known by the consumer [M8.23]. The comparison of the color name specified by the DocumentImpositionColor PrintTicket setting with the colorant name in the profile's colorantTable MUST be performed as a case-sensitive ASCII comparison after trimming leading and trailing whitespace from each string [M8.24].

The imposition named color is used *only* to compute XYZ values for consumers that do not perform separation. For consumers that do perform separation, it is an indicator that the tint level supplied SHOULD be used for all device colorants [S8.7]. Producers SHOULD create the profile used by the imposition color in such a way that it does not lay down excessive ink when printed on a device that does not perform separation and uses the profile to compute XYZ values instead [S8.8].

15.4 Alpha and Gradient Blending with Rich Colors

For consumers that understand rich colors, it is necessary to understand how they may be blended to create gradient or transparency effects. The PrintTicket specifies the color space that SHOULD be used for blending gradients and transparencies in the PageBlendColorSpace setting [S8.9]. These settings apply to the page level.

If a consumer understands the PageBlendColorSpace PrintTicket setting, it SHOULD convert all color to the specified blending color space before performing a blend operation [S8.9]. For gradients, the specified blending color space is used only if no gradient stop color values are specified using sRGB or scRGB colors. If any of the gradient stop color values are specified using sRGB or scRGB colors or the consumer does not understand the PageBlendColorSpace PrintTicket setting, the color interpolation mode of the gradient brush MUST be used instead [M8.25].

The behavior of documents using rich color features is implementation specific. Consumers MUST support sRGB [M8.1], but they MAY support rich color spaces such as scRGB or CMYK as well [O8.6]. Consumers that encounter any document using non-sRGB colors MAY process those colors using the simpler sRGB color space, resulting in deviations, especially for alpha blending [O8.6].

15.5 PrintTicket Color Settings

This section summarizes the color-related PrintTicket settings. For more information, refer to the Print Schema.

Table 15–3. PrintTicket color settings

Feature or ParameterDef	Option/ScoredProperty or Properties	Description
PageColorManagement	Device (default)	Perform color management only in device.
	Driver	Allow driver to perform color management. The driver MAY color manage elements or convert them to different color spaces [O8.7].
PageDeviceColorSpaceProfileURI	*profileUri properties*	Identifies an ICC profile contained in the XPS package. The processing of this option depends of the setting of the PageDeviceColorSpaceUsage feature. No default value is specified. Contains an absolute part name relative to the package root. All elements using that profile are assumed to be already in the appropriate device color space, and will not be color managed in the driver or device.
PageDeviceColorSpaceUsage	MatchToDefault (default)	If the device determines that the profile specified by the PageDeviceColorSpaceProfileURI feature can be used as a device color space profile, all elements using the same profile are treated as already being specified in device color space. However, the device's internal color profile SHOULD be used for color management of all other elements [S8.10].

		If the profile cannot be used as a device color space profile, elements using the profile MUST be color managed like any other element using a color profile [M8.27].
	OverrideDeviceDefault	If the profile specified by the PageDeviceColorSpaceProfileURI parameter definition has a number of channels matching the number of primaries of the device, it SHOULD be used instead of the device's internal color management for all elements [S8.11]. Elements using this profile are assumed to be in device color space and will not be color managed further.
PageBlendColorSpace	sRGB (default)	The sRGB color space that SHOULD be used for blending [S8.9].
	scRGB	The scRGB color space that SHOULD be used for blending [S8.9].
	ICCProfile	The Uri property of the option specifies an ICC profile defining the color space that SHOULD be used for blending [S11.13]. The Uri is an absolute part name relative to the package root. The profile MUST be an output profile (containing AToB0Tag, BToA0Tag, AToB1Tag, BToA1Tag, AToB2Tag, and BToA2Tag), otherwise it MUST be ignored [M8.28]. The rendering intent specified by PageICMRenderingIntent PrintTicket setting is used, unless the profile specifies a rendering intent of its own.

		Elements using the profile specified by PageBlendColorSpace MAY be blended naively (channel-by-channel) without converting through PCS [O8.8].
PageICMRenderingIntent	AbsoluteColorimetric RelativeColorimetric (default) Photographs BusinessGraphics	The rendering intent as defined by the ICC Version 2 specification. This value SHOULD be ignored for elements using a profile that specifies the rendering intent in the profile [S8.13].
DocumentImpositionColor	*colorName properties*	Elements using the named color identified by the colorName properties MUST appear on all color separations [M8.29]. See §15.3, "Color Separation," on page 217 and §15.1.16, "Named Color Syntax," on page 212 for details. Consumers that do not produce separations treat these elements like other elements using named color, without any additional required processing steps. No default value is specified.
PageBlackGenerationProcessing	Automatic	Default.
	Custom/TotalInkCoverageLimit	The maximum allowable sum of the four ink coverages anywhere in an image or element (200% to 400%).
	Custom/BlackInkLimit	The maximum allowed K-channel value (0% to 100%).
	Custom/GrayComponentReplacementLevel	The percentage of gray component replacement to perform (0% to 100%).
	Custom/GrayComponentReplacementStart	The point in the highlight-to-shadow range where GCR should start (0% to

		100%; 100% = darkest shadow).
	Custom/ GrayComponentReplacement Extent	The extent beyond neutrals (into chromatic colors) that GCR applies. 0% = Undercolor component replacement; 100% = Gray component replacement.
	Custom/UnderColorAdditionStart	The shadow level below which UCA will be applied (0% to 100%).
	Custom/UnderColorAdditionLevel	The amount of chromatic ink (in gray component ratios) to add to areas where GCR/UCR has generated “BlackInkLimit” (or “UCAStart”, if specified) in the dark neutrals and near-neutral areas (0% to 100%).

16. Document Structure and Interactivity

Some consumers support enhanced interactive functionality through features such as text selection, navigation, and hyperlinking. Others, such as screen readers, provide enhanced accessibility. These features rely on structural information beyond what can be inferred from the page markup. Producers can author this information explicitly.

The methods for adding document structure described here are OPTIONAL [O9.1]. Consumers MAY ignore any authored document structure or hyperlinks [O9.1], particularly where they are not relevant (such as in the case of printers). Recommended consumer behavior in the absence of document structure information is also described.

Document structure is defined with markup in the FixedPage, FixedDocument, DocumentStructure, and StoryFragments parts.

16.1 Document Structure Markup

Document structure markup consists of two structural concepts. The first is the *document outline*, which contains a structured list of indices into the XPS Document, similar to a table of contents. The second is the *document content*, which identifies blocks of individually readable content. These blocks are called *stories*.

A story can extend across multiple pages, and several stories can share a single page. A story can include the entire contents of an XPS Document, or it can include only an individual block of readable content, such as a single newspaper article. Like a newspaper article, the story may appear in blocks throughout the XPS Document. [*Example*: The first part could appear on page 1 and the second part on page 5. *end example*] Since a story can span multiple pages, the document content identifies which FixedPage parts contain fragments of a particular story.

A *story fragment* is the portion of a story that appears within a single fixed page. The story fragment contains the structural markup for all text and images related to a particular story on a particular page. When a producer specifies the document structure, every FixedPage part has a corresponding StoryFragments part that contains all of the story fragments for that page.

Each story fragment contains content structure information. *Content structure* is the set of markup elements that allow expression of well-understood semantic blocks, such as paragraphs, tables, lists, and figures. Content structure markup enables features such as paragraph and table selection, screen reading, and rich-format copying.

Producers MAY provide either the document outline or the document content, or both; consumers MAY ignore either or both [O9.2].

16.1.1 DocumentStructure Part

The fundamental building block of document structure markup is the named element. A *named element* refers to an element in the fixed page markup with a specified **Name** attribute. Every meaningful element in the fixed page markup SHOULD specify a **Name** attribute in order for the document structure markup to refer to it [S9.1].

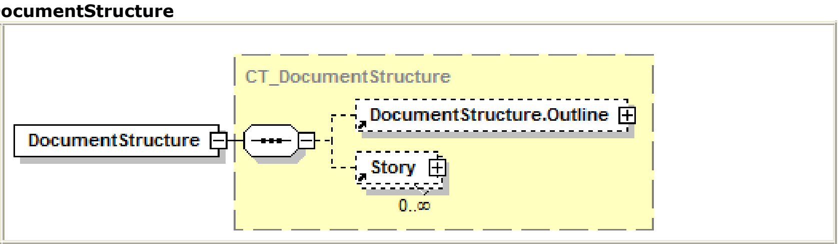
Document structure markup SHOULD NOT refer to a single named element more than once in the document content or to a named element that embeds another named element that it also refers to. When referring to a <Canvas> element, producers SHOULD consider all descendant elements to be referenced in markup order [S9.3]. Consumers MAY choose to interpret these scenarios as duplicate document content [O9.3].

Children of <VisualBrush> elements SHOULD NOT be referenced by document structure markup [S9.30].

Because each named element in a FixedPage part that is intended as an addressable location is specified in the <PageContent.LinkTargets> element in the FixedDocument part, consumers MAY first attempt to locate named elements directly from the FixedDocument part [O9.4].

16.1.1.1 <DocumentStructure> Element

element **DocumentStructure**

diagram	
annotation	The root element of the DocumentStructure part.

The <DocumentStructure> element is the root element of the DocumentStructure part. It MAY contain a single <DocumentStructure.Outline> element and zero or more <Story> elements [M2.72].

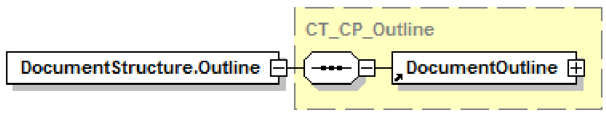
Example 16–1. Document structure markup

```
<DocumentStructure
   xmlns="http://schemas.microsoft.com/xps/2005/06/documentstructure">
   <DocumentStructure.Outline>
      ...
   </DocumentStructure.Outline>
   <Story>
      ...
   </Story>
   <Story>
      ...
   </Story>
</DocumentStructure>
```

end example]

16.1.1.2 <DocumentStructure.Outline> Element

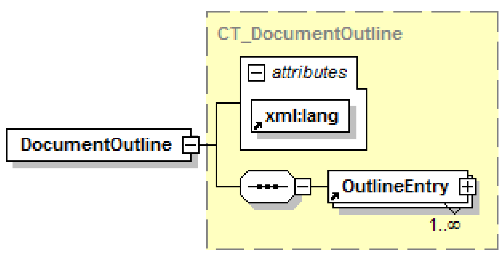
element **DocumentStructure.Outline**

diagram	
annotation	Contains a structured document outline that provides a list of links into the document contents or external sites.

The <DocumentStructure.Outline> element is the root element of the document outline. The <DocumentStructure.Outline> element contains only a single <DocumentOutline> element.

16.1.1.3 <DocumentOutline> Element

element **DocumentOutline**

diagram	
attributes	Name: xml:lang; Type: ; Use: required; Default: ; Fixed: ; Annotation: This attribute specifies the default language used for any child element contained within the current element or nested child elements. The language is specified according to IETF RFC 3066.
annotation	Specifies a list of meaningful indices into the XPS Document, similar to a table of contents, or to external URIs, such as web addresses.

Attributes:

Name	Type	Use	Default	Fixed	Annotation
xml:lang		required			This attribute specifies the default language used for any child element contained within the current element or nested child elements. The language is specified according to IETF RFC 3066.

The <DocumentOutline> element lets producers specify an organizational hierarchy in the form of a list of URIs to locations in the fixed page markup or to external addresses, similar to a table of contents or a set of bookmarks. The <DocumentOutline> element contains only <OutlineEntry> elements.

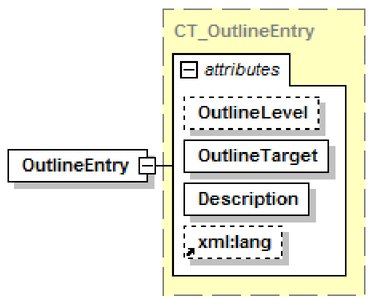
The **xml:lang** attribute specifies the default language used by the **Description** attribute of the child <OutlineEntry> element.

Consumers can use the document outline to implement such features as a table of contents or a navigation pane.

16.1.1.4 <OutlineEntry> Element

element **OutlineEntry**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
OutlineLevel	ST_IntGEOne	optional	1		A description of the level where the outline entry exists in the hierarchy. A value of 1 is the root.
OutlineTarget	xs:anyURI	required			The URI to which the outline entry is linked. This may be a URI to a named element within the document or an external URI, such as a website. It can be used as a hyperlink destination.
Description	xs:string	required			The friendly text associated with this outline entry.
xml:lang		optional			This attribute specifies the default language used for any child element contained within the current element or nested child elements. The language is specified according to IETF RFC 3066.

annotation: Represents an index to a specific location in the document.

Each <OutlineEntry> element represents an index to a specific location in the document or a specific location external to the document. Consumers can use the document outline information to support interactive functionality.

Example 16–2. Document outline markup

A viewing consumer can create a navigation pane that uses the Unicode value of the **Description** attribute of each <OutlineEntry> element. The corresponding location is specified by the **OutlineTarget** attribute, which are specified in a manner identical to hyperlinks. The **OutlineLevel** attribute allows consumers to indent entries in the navigation pane.

```
<DocumentStructure
   xmlns="http://schemas.microsoft.com/xps/2005/06/documentstructure">
   <DocumentStructure.Outline>
      <DocumentOutline>
         <OutlineEntry
            OutlineLevel="1"
            Description="1. Documents"
            OutlineTarget="../FixedDoc.fdoc#Documents_1" />
         <OutlineEntry
            OutlineLevel="2"
            Description="1.1. Paragraphs"
            OutlineTarget="../FixedDoc.fdoc#Paragraphs_1_1" />
      </DocumentOutline>
   </DocumentStructure.Outline>
</DocumentStructure>
```

A consumer might display this information as follows, with the first entry linked to `Documents_1` and the second entry linked to `Paragraphs_1_1`.

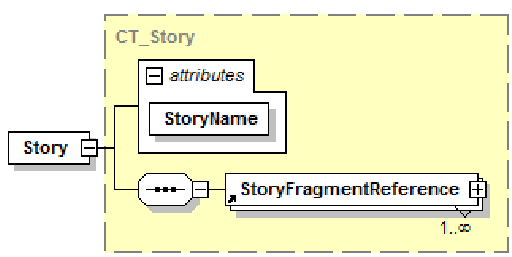
```
1. Documents
   1.1. Paragraphs
```

end example]

16.1.1.5 <Story> Element

element **Story**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
StoryName	xs:string	required			The name used by story fragments to identify they belong to this story.

annotation

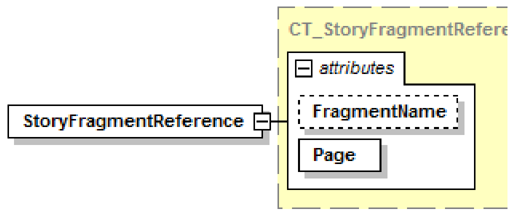
Defines a single story and where each of its story fragments appear in the XPS Document.

The <Story> element is the root for a single story and orders all of the story fragments containing content structure information such as sections, paragraphs, and tables. Each story has a unique name that is used to correlate the content structure for each page to that story. The <Story> element contains one or more <StoryFragmentReference> elements.

16.1.1.6 <StoryFragmentReference> Element

element **StoryFragmentReference**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
FragmentName	xs:string	optional			Used to distingush between multiple story fragments from the same story on a single page.
Page	ST_IntGEOne	required			Identifies the page number of the document that the story fragment is related to. Page numbers start at 1 and correspond to the order of <PageContent> elements in the FixedDocument part.

annotation

Identifies the StoryFragments part where this individual story fragment is defined.

The <StoryFragmentReference> element identifies the page with a relationship to the StoryFragments part in which the single story fragment is defined. By identifying where in the XPS Document each story fragment appears, consumers can easily access only the pages that contain a particular story.

Each page that contains a story fragment is identified by number. This number refers to the *n*th page of the XPS Document referenced within the fixed document sequence and fixed document markup, starting at the fixed payload root. This value is identified in the **Page** attribute. The StoryFragments part containing the corresponding content structure is referenced by retrieving the part associated via relationship from the indicated page. This allows consumers to access only the pages of the document that contain the story of interest. It is also possible for a single story to return to a page containing a different fragment of the same story.

The **FragmentName** attribute MUST be unique within the scope of the story [M2.72].

Example 16–3. Simple multi-story document

The following markup describes a four-page document containing one story that covers the first one and one-half pages and then continues on page 4. It is interrupted by a second story that begins in the middle of page 2 and concludes on page 3.

```
<DocumentStructure
   xmlns="http://schemas.microsoft.com/xps/2005/06/documentstructure">
   <Story Name="Story1">
      <StoryFragmentReference Page="1"/>
      <StoryFragmentReference Page="2"/>
      <StoryFragmentReference Page="4"/>
   </Story>
   <Story Name="Story2">
      <StoryFragmentReference Page="2"/>
      <StoryFragmentReference Page="3"/>
   </Story>
</DocumentStructure>
```

end example]

Example 16–4. Story flowing back and forth across a page boundary

The following markup describes a page containing two tables, arranged side-by-side, each of which continues to the following page. In this case, the fragment is split and a fragment name is specified. `FragmentA` refers to the content leading up to the middle of the first (left) table and `FragmentB` is the continuation of this table on the following page. The flow then returns to the second (right) table on page 1 (`FragmentC`) before continuing with the rest of the story in `FragmentD`.

```
<DocumentStructure
   xmlns="http://schemas.microsoft.com/xps/2005/06/documentstructure">
   <Story Name="Story_1">
      <StoryFragmentReference FragmentName="FragmentA" Page="1"/>
      <StoryFragmentReference FragmentName="FragmentB" Page="2"/>
      <StoryFragmentReference FragmentName="FragmentC" Page="1"/>
      <StoryFragmentReference FragmentName="FragmentD" Page="2"/>
   </Story>
</DocumentStructure>
```

end example]

16.1.2 StoryFragments Part

The StoryFragments part contains content structure markup (describing such things as tables and paragraphs) for each story fragment that appears on the page. The content structure is expressed by tags that ultimately wrap <NamedElement> references that point to fixed page markup.

Table 16–1. StoryFragments part elements

Name	Description
<StoryFragments>	Root element.
<StoryFragment>	Contains all content structure markup elements for a single story fragment.
<StoryBreak>	Presence of this element indicates that the following or preceding markup is not continued to the previous or

	next story fragment, depending on whether the element is at the beginning or end of the story fragments markup.
<SectionStructure>	Arbitrary structural grouping element.
<TableStructure>	Contains a full table definition.
<TableRowGroupStructure>	Contains a group of table rows.
<TableRowStructure>	Contains a row of table cells.
<TableCellStructure>	Contains structural elements representing the contents of a table cell.
<ListStructure>	Group of related items.
<ListItemStructure>	Individual item in a list.
<FigureStructure>	Group of related named elements that should be interpreted as a whole (such as a diagram).
<ParagraphStructure>	Group of named elements that constitute a paragraph.
<NamedElement>	Element that links the document structure markup to the fixed page markup.

Because a single content structural element may be split across pages, the <StoryBreak> element is provided to identify that a given element continues *to* the next story fragment or continues *from* a previous story fragment. A <StoryBreak> element MUST NOT be included in a position other than the first or last child element of a <StoryFragment> element [M2.72].

If a <StoryBreak> element is not present at the beginning of the content structure markup, consumers SHOULD consider the markup a continuation of the previous story fragment that must be merged [S9.4]. Likewise, if a <StoryBreak> element is not present at the end of the content structure markup, consumers SHOULD consider the markup a continuation to the next story fragment that must be merged to determine the cross-fragment content structure [S9.4].

Content structure is merged on an element-by-element basis, merging the last element closed in the leading story fragment with the first element opened in the trailing story fragment. This process continues until the closing tag from the leading story fragment no longer matches the opening tag from the trailing story fragment.

<TableCellStructure> elements require special merging, such that all <TableCellStructure> elements within a <TableRowStructure> element are merged. In order to merge the table cells and rows correctly, producers MUST specify empty <TableCellStructure> elements for cells that do not break across story fragments [M9.1].

Example 16–5. Content structure spanning pages

Given the following two StoryFragments parts, consumers may construct the content structure as shown.

```
<!-- First StoryFragments part -->

<StoryFragments
   xmlns="http://schemas.microsoft.com/xps/2005/06/documentstructure">
   <StoryFragment FragmentType="Header">
      <StoryBreak />
      <ParagraphStructure>
         <NamedElement NameReference="Block1" />
      </ParagraphStructure>
      <StoryBreak />
   </StoryFragment>
   <StoryFragment StoryName="Story1" FragmentType="Content">
      <StoryBreak />
      <SectionStructure>
         <TableStructure>
            <TableRowGroupStructure>
               <TableRowStructure>
                  <TableCellStructure>
                     <ParagraphStructure>
                        <NamedElement NameReference="Block2" />
                        <NamedElement NameReference="Block3" />
                     </ParagraphStructure>
                  </TableCellStructure>
                  <TableCellStructure>
                     <ParagraphStructure>
                        <NamedElement NameReference="Block4" />
                     </ParagraphStructure>
                  </TableCellStructure>
               </TableRowStructure>
               <TableRowStructure>
                  <TableCellStructure>
                     <ParagraphStructure>
                        <NamedElement NameReference="Block5" />
                        <NamedElement NameReference="Block6" />
                     </ParagraphStructure>
                  </TableCellStructure>
                  <TableCellStructure>
                     <ParagraphStructure>
                        <NamedElement NameReference="Block7" />
                     </ParagraphStructure>
                  </TableCellStructure>
               </TableRowStructure>
            </TableRowGroupStructure>
         </TableStructure>
      </SectionStructure>
   </StoryFragment>
   <StoryFragment FragmentType="Footer">
      <StoryBreak />
      <ParagraphStructure>
         <NamedElement NameReference="Block8" />
      </ParagraphStructure>
      <StoryBreak />
   </StoryFragment>
</StoryFragments>
```

```
<!-- Second StoryFragments part -->

<StoryFragments
   xmlns="http://schemas.microsoft.com/xps/2005/06/documentstructure">
   <StoryFragment FragmentType="Header">
      <StoryBreak />
      <ParagraphStructure>
         <NamedElement NameReference="Block9" />
      </ParagraphStructure>
      <StoryBreak />
   </StoryFragment>
   <StoryFragment StoryName="Story1" FragmentType="Content">
      <SectionStructure>
         <TableStructure>
            <TableRowGroupStructure>
               <TableRowStructure>
                  <TableCellStructure />
                  <TableCellStructure>
                     <ParagraphStructure>
                        <NamedElement NameReference="Block10" />
                        <NamedElement NameReference="Block11" />
                     </ParagraphStructure>
                  </TableCellStructure>
               </TableRowStructure>
               <TableRowStructure>
                  <TableCellStructure>
                     <ParagraphStructure>
                        <NamedElement NameReference="Block12" />
                     </ParagraphStructure>
                  </TableCellStructure>
                  <TableCellStructure>
                     <ParagraphStructure>
                        <NamedElement NameReference="Block13" />
                     </ParagraphStructure>
                  </TableCellStructure>
               </TableRowStructure>
            </TableRowGroupStructure>
         </TableStructure>
      </SectionStructure>
      <StoryBreak />
   </StoryFragment>
   <StoryFragment FragmentType="Footer">
      <StoryBreak />
      <ParagraphStructure>
         <NamedElement NameReference="Block14" />
      </ParagraphStructure>
      <StoryBreak />
   </StoryFragment>
</StoryFragments>
```

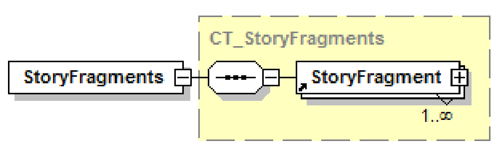
```
<!-- Resulting merged content structure for Story1 -->

<SectionStructure>
   <TableStructure>
      <TableRowGroupStructure>
         <TableRowStructure>
            <TableCellStructure>
               <ParagraphStructure>
                  <NamedElement NameReference="Block2" />
                  <NamedElement NameReference="Block3" />
               </ParagraphStructure>
```

```
            </TableCellStructure>
            <TableCellStructure>
               <ParagraphStructure>
                  <NamedElement NameReference="Block4" />
               </ParagraphStructure>
            </TableCellStructure>
         </TableRowStructure>
         <TableRowStructure>
            <TableCellStructure>
               <ParagraphStructure>
                  <NamedElement NameReference="Block5" />
                  <NamedElement NameReference="Block6" />
               </ParagraphStructure>
            </TableCellStructure>
            <TableCellStructure>
               <ParagraphStructure>
                  <NamedElement NameReference="Block7" />
                  <NamedElement NameReference="Block10" />
                  <NamedElement NameReference="Block11" />
               </ParagraphStructure>
            </TableCellStructure>
         </TableRowStructure>
         <TableRowStructure>
            <TableCellStructure>
               <ParagraphStructure>
                  <NamedElement NameReference="Block12" />
               </ParagraphStructure>
            </TableCellStructure>
            <TableCellStructure>
               <ParagraphStructure>
                  <NamedElement NameReference="Block13" />
               </ParagraphStructure>
            </TableCellStructure>
         </TableRowStructure>
      </TableRowGroupStructure>
   </TableStructure>
</SectionStructure>
```

end example]

16.1.2.1 <StoryFragments> Element

element **StoryFragments**

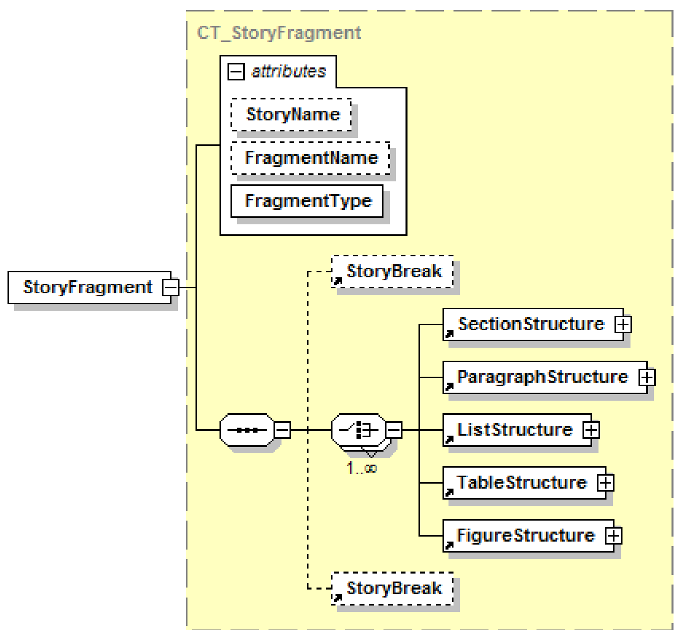
diagram	
annotation	The root of a StoryFragments part. Contains all story fragments that appear on a specific page.

The <StoryFragments> element groups all of the <StoryFragment> elements on a page.

16.1.2.2 <StoryFragment> Element

element **StoryFragment**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
StoryName	xs:string	optional			Identifies the story that this story fragment belongs to. If omitted, the story fragment is not associated with any story.
FragmentName	xs:string	optional			Used to uniquely identify the story fragment.
FragmentType	ST_FragmentType	required			Specifies the type of content included in the story fragment. Valid values are Content, Header, and Footer.

annotation

Specifies the document structural markup that appears on the current page for a single story block.

Each <StoryFragment> has a **StoryName** attribute that associates it with a story defined in the DocumentStructure part. It also has a **FragmentType** attribute, the values for which are Content (the default), Header, or Footer.

Headers and footers are defined in their own story fragment on each page. These stories do not specify a **StoryName** value, so they are essentially unreferenced stories that exist only on a single page.

Producers authoring document structure information SHOULD reference every element of the fixed page markup that has semantic meaning (such as text or images) in the StoryFragments parts [S9.5].

Example 16–6. StoryFragments part markup

The following markup describes the StoryFragments part of a one-page document:

```
<StoryFragments
   xmlns="http://schemas.microsoft.com/xps/2005/06/documentstructure">
   <StoryFragment FragmentType="Header">
      <StoryBreak />
      <ParagraphStructure>
         <NamedElement NameReference="Block13" />
         <NamedElement NameReference="Block14" />
      </ParagraphStructure>
      <StoryBreak />
   </StoryFragment>
   <StoryFragment StoryName="Story1" FragmentType="Content">
      <StoryBreak />
      <ParagraphStructure>
         <NamedElement NameReference="Block1" />
         <NamedElement NameReference="Block2" />
      </ParagraphStructure>
      <TableStructure>
         <TableRowGroupStructure>
            <TableRowStructure>
               <TableCellStructure>
                  <ParagraphStructure>
                     <NamedElement NameReference="Block3" />
                     <NamedElement NameReference="Block4" />
                  </ParagraphStructure>
               </TableCellStructure>
               <TableCellStructure>
                  <ParagraphStructure>
                     <NamedElement NameReference="Block5" />
                  </ParagraphStructure>
               </TableCellStructure>
            </TableRowStructure>
         </TableRowGroupStructure>
      </TableStructure>
      <SectionStructure>
         <ParagraphStructure>
            <NamedElement NameReference="Block6" />
         </ParagraphStructure>
         <ParagraphStructure>
            <NamedElement NameReference="Block7" />
            <NamedElement NameReference="Block8" />
         </ParagraphStructure>
      </SectionStructure>
      <SectionStructure>
         <FigureStructure>
            <NamedElement NameReference="Block9" />
         </FigureStructure>
```

```
         <ListStructure>
            <ListItemStructure>
               <ParagraphStructure>
                  <NamedElement NameReference="Block10" />
               </ParagraphStructure>
            </ListItemStructure>
            <ListItemStructure>
               <ParagraphStructure>
                  <NamedElement NameReference="Block11" />
               </ParagraphStructure>
            </ListItemStructure>
            <ListItemStructure>
               <ParagraphStructure>
                  <NamedElement NameReference="Block12" />
               </ParagraphStructure>
            </ListItemStructure>
         </ListStructure>
      </SectionStructure>
      <StoryBreak />
   </StoryFragment>
   <StoryFragment FragmentType="Footer">
      <StoryBreak />
      <ParagraphStructure>
         <NamedElement NameReference="Block15" />
         <NamedElement NameReference="Block16" />
         <NamedElement NameReference="Block17" />
      </ParagraphStructure>
      <StoryBreak />
   </StoryFragment>
</StoryFragments>
```

end example]

A <StoryFragment> element MAY be identified with a FragmentName attribute to distinguish it from other fragments for the same story on a single page [M2.72].

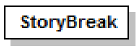
Example 16–7. Story fragments markup using a fragment name

```
<StoryFragments
   xmlns="http://schemas.microsoft.com/xps/2005/06/documentstructure">
   <StoryFragment
      StoryName="Story1"
      FragmentName="Fr1"
      FragmentType="Content">
      <StoryBreak />
      <ParagraphStructure>
         <NamedElement NameReference="Block1" />
         <NamedElement NameReference="Block2" />
      </ParagraphStructure>
      <StoryBreak />
   </StoryFragment>
   <StoryFragment
      StoryName="Story1"
      FragmentName="Fr2"
      FragmentType="Content">
      <StoryBreak />
      <ParagraphStructure>
         <NamedElement NameReference="Block8" />
      </ParagraphStructure>
      <StoryBreak />
   </StoryFragment>
</StoryFragments>
```

end example]

16.1.2.3 <StoryBreak> Element

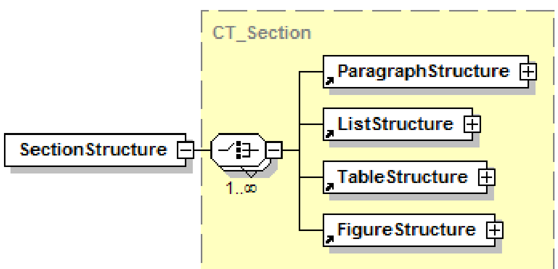
element **StoryBreak**

diagram	
annotation	If located at the beginning of a <StoryFragment> definition, indicates that the following markup elements should not be merged with the markup from the previous <StoryFragment>. If located at the end of a <StoryFragment> definition, indicates that the preceding markup elements should not be merged with the subsequent <StoryFragment>.

The <StoryBreak> element signals to the consumer not to perform merging across story fragments to determine the content structure.

16.1.2.4 <SectionStructure> Element

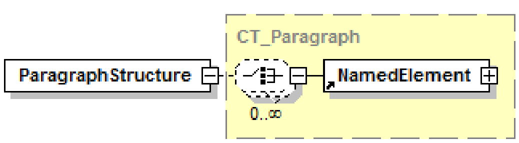
element **SectionStructure**

diagram	
annotation	Provides an arbitrary grouping of content structural markup elements.

The <SectionStructure> element provides an arbitrary grouping of <Paragraph>, <TableStructure>, <ListStructure>, and <FigureStructure> elements.

16.1.2.5 <ParagraphStructure> Element

element **ParagraphStructure**

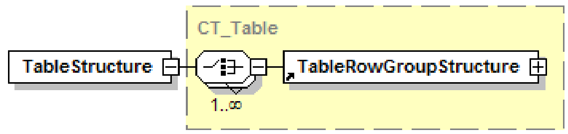
diagram	
annotation	Contains the named elements that constitute a single paragraph.

A <ParagraphStructure> element describes the list of <NamedElement> elements that constitute a single paragraph.

16.1.2.6 <TableStructure> Element

element **TableStructure**

diagram



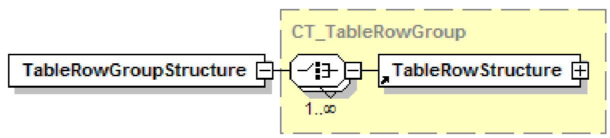
annotation: Contains a complete definition of a table in the XPS Document.

A <TableStructure> element is the complete definition of a table. An implementation MAY use it to build special functionality, such as row or column selection [O9.5].

16.1.2.7 <TableRowGroupStructure> Element

element **TableRowGroupStructure**

diagram



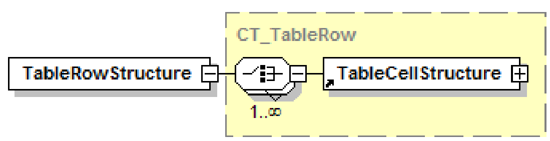
annotation: Contains the set of table rows that make up a table.

A <TableRowGroupStructure> element is REQUIRED in order to specify a set of <TableRowStructure> elements [M2.72].

16.1.2.8 <TableRowStructure> Element

element **TableRowStructure**

diagram



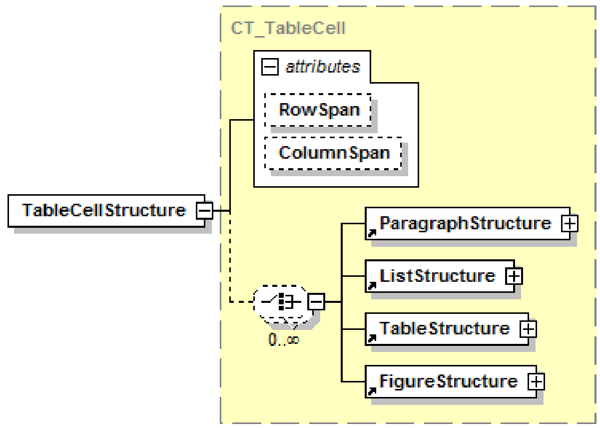
annotation: Contains the set of table cells that make up a row of a table.

This element groups <TableCellStructure> child elements that define a single row of a table.

16.1.2.9 <TableCellStructure> Element

element **TableCellStructure**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
RowSpan	ST_TableSpan	optional	1		Indicates the number of rows this cell spans, or merges into a single cell.
ColumnSpan	ST_TableSpan	optional	1		Indicates the number of columns this cell spans, or merges into a single cell.

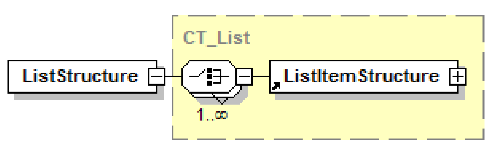
annotation: Contains the elements that occupy a single cell of a table.

This element defines the appearance of a table cell. It MAY contain nested <TableStructure> elements [M2.72].

16.1.2.10 <ListStructure> Element

element **ListStructure**

diagram

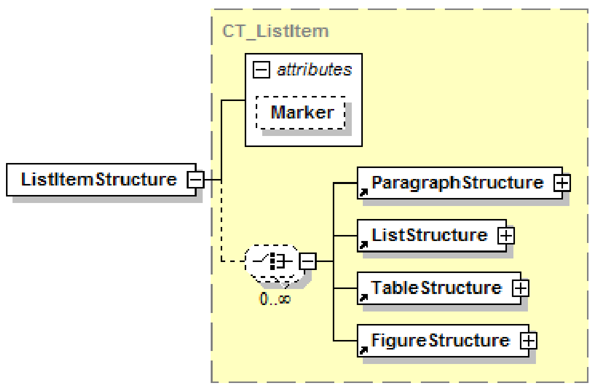


annotation: Contains a collection of items that are group together in a list.

The <ListStructure> element is the complete definition of a list of related items.

16.1.2.11 <ListItemStructure> Element

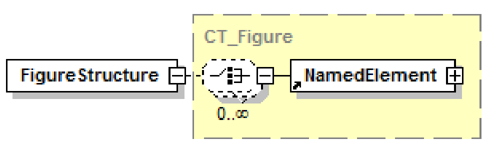
element **ListItemStructure**

<table>
<tr><td>diagram</td><td>
</td></tr>
<tr><td>attributes</td><td><table><tr><td>Name</td><td>Type</td><td>Use</td><td>Default</td><td>Fixed</td><td>Annotation</td></tr><tr><td>Marker</td><td>ST_NameUnique</td><td>optional</td><td></td><td></td><td>The named element that represents the marker for this list items, such as a bullet, number, or image.</td></tr></table></td></tr>
<tr><td>annotation</td><td>Describes a single structural block. These structural blocks are grouped together in a list.</td></tr>
</table>

A <ListItemStructure> element defines a single item in a list.

16.1.2.12 <FigureStructure> Element

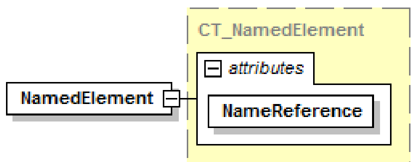
element **FigureStructure**

<table>
<tr><td>diagram</td><td>
</td></tr>
<tr><td>annotation</td><td>Groups the named elements that constitute a single drawing or diagram.</td></tr>
</table>

A <FigureStructure> element includes a group of named elements that comprise a single drawing or diagram.

16.1.2.13 <NamedElement> Element

element **NamedElement**

<table>
<tr><td>diagram</td><td>
</td></tr>
<tr><td>attributes</td><td>
<table>
<tr><td>Name</td><td>Type</td><td>Use</td><td>Default</td><td>Fixed</td><td>Annotation</td></tr>
<tr><td>NameReference</td><td>ST_Name</td><td>required</td><td></td><td></td><td>Identifies the named element in the FixedPage part markup that is referenced by the document structure markup.</td></tr>
</table>
</td></tr>
<tr><td>annotation</td><td>All document structure is related to the fixed page markup using this element. The <NamedElement> points to a single markup element contained in the fixed page markup.</td></tr>
</table>

A <NamedElement> references a specific element in the fixed page by using the **NameReference** attribute to specify an element in the fixed page markup with a corresponding name.

If the targeted fixed page uses markup compatibility markup that changes the presence of certain named elements, the StoryFragments part should also use it in order to reference each element in either representation.

16.2 Hyperlinks

If consumers enable user interactivity, they SHOULD support hyperlink activation and addressing [S9.6].

16.2.1 Hyperlink Activation

Hyperlinks are specified inline on any <Canvas>, <Path>, or <Glyphs> element by means of the **FixedPage.NavigateUri** attribute. The value of the attribute is the destination URI. If hyperlinked <Path> or <Glyphs> elements are rendered as overlapping on the page, consumers MUST treat the topmost element as the only hyperlink that may be activated in the overlapping region [M9.2].

When activating a hyperlink, consumers SHOULD load the specified resource if they understand the URI type. If the URI is an internal reference to the XPS Document, consumers SHOULD navigate to the URI [S9.7].

If a producer specifies a **FixedPage.NavigateUri** attribute on a <Canvas> element, consumers MUST treat all child elements of that canvas as having an associated hyperlink [M9.3]. Child or descendant elements can override this value with their own **FixedPage.NavigateUri** attribute.

Relative internal hyperlinks between FixedPage parts MUST specify, at a minimum, the named address relative to the FixedDocument part [M9.4].

Producers can mark any <FixedPage>, <Canvas>, <Path>, or <Glyphs> element as an addressable location within the XPS Document by specifying a value for the **Name** attribute. The name SHOULD be unique within the scope of the fixed document [S9.8]. If it is not unique, only the first occurrence of the named address is addressable.

These elements, if specified as a <VisualBrush.Visual> property element are not addressable by a hyperlink.

It is RECOMMENDED that **Name** attribute values be unique within an entire fixed document sequence [S9.9]. If they are not, only the first occurrence of the named address is addressable from an external location. Internal hyperlinks can specify a named element fragment relative to a particular fixed document, but consumers MAY interpret such a URI relative to the entire fixed document sequence instead [O9.6].

In order to be addressable by either a hyperlink or the document outline, the named address MUST appear in the <PageContent.LinkTargets> element in the fixed document [M9.5]. If a named address appears in the <PageContent.LinkTargets> element in the fixed document but is not found in the **Name** attribute of an element within the associated fixed page, consumers MUST treat the top of the associated fixed page as the named address [M9.6]. If the named address in a URI fragment is not found, consumers MUST ignore the fragment portion of the URI [M9.7].

Example 16–8. A relative, internal, named-address hyperlink

```
FixedPage.NavigateUri="../MyDocument.fdoc#MyAddress"
```

end example]

16.2.2 Hyperlink Addressing

XPS Documents specify two forms of URI fragment identifiers to address locations within an XPS Document. The first is a named address. [*Example*: "http://xps/MyPackage#MyAddress", where "http://xps/MyPackage" is an XPS Document and "MyAddress" is a named address within the document. *end example*] The second is an absolute page number within the XPS Document. [*Example*: "http://xps/MyPackage#15", where "15" references the FixedPage part associated with the fifteenth <PageContent> entry among all the fixed documents in the fixed document sequence. *end example*]

Page number fragment identifiers refer to the absolute page number (1-based) in the fixed document sequence. [*Example*: If an XPS Document has a 3-page fixed document, followed by a 10-page fixed document, followed by an 8-page fixed document, the fragment identifier "#15" refers to the second page of the third fixed document in the fixed document sequence. *end example*] Internal references MUST specify a page address relative to the fixed document sequence [M9.8].

Example 16–9. A relative internal page address hyperlink

```
FixedPage.NavigateUri="../../../MyDocSeq.fdseq#12"
```

end example]

16.2.3 Name Attribute

The **Name** attribute contains a string value that identifies the current element as a named, addressable point for the purpose of hyperlinking. The **Name** attribute is optional. Names SHOULD be unique within a fixed document [S9.8], and it is RECOMMENDED that they be unique within a fixed document sequence [S9.9]. The **Name** attribute MUST NOT be specified on any children of a <ResourceDictionary> element [M9.10].

If the **Name** attribute is specified, producers SHOULD also create a corresponding <LinkTarget> element in the FixedDocument part within the <PageContent> element that links to the parent fixed page [S9.10]. Consumers MAY ignore this attribute [O9.7], but devices that support user interaction with the contents of XPS Documents SHOULD support hyperlinks [S9.6].

The **Name** value, if specified, MUST meet the following requirements [M2.72]:

1. The initial character MUST be an underscore character or a letter, that is, it falls within the Lu, Ll, Lo, Lt, and Nl categories [M2.72].
2. Trailing characters MUST be an underscore character or a letter or number, that is, they fall within the Lu, Ll, Lo, Lt, Nl, Mn, Mc, and Nd categories [M2.72].

[*Note*: These requirements match those of XML identifiers with additional restrictions. *end note*]

The category abbreviations, as defined within the Unicode Character Database, are partially reproduced in Table 16–2.

Table 16–2. Unicode character categories

Abbreviation	Description
Lu	Letter, uppercase
Ll	Letter, lowercase
Lt	Letter, titlecase
Lo	Letter, other
Mn	Mark, non-spacing
Mc	Mark, spacing combining
Nd	Number, decimal
Nl	Number, letter

16.2.4 FixedPage.NavigateUri Attribute

The **FixedPage.NavigateUri** attribute associates a hyperlink URI with an element, making it a hyperlink source. Its value may be a relative or absolute URI that addresses a resource that is internal or external to the XPS Document package, respectively. The base URI used to resolve a relative URI is that of the FixedPage part in which the element with the **FixedPage.NavigateUri** attribute appears. Therefore, a hyperlink to a destination within the fixed document of the source MUST specify the destination in the context of the FixedDocument part [M9.4]. [*Example*: "../FixedDoc1.fdoc#MyDestination". *end example*] A destination in the same fixed document SHOULD be expressed as a relative URI [S9.11].

The **FixedPage.NavigateUri** attribute is OPTIONAL [M2.72]. It SHOULD be included *only* if the element is intended to be a hyperlink. Consumers MAY ignore this attribute [O9.8], but devices that support user interaction with the contents of XPS Documents SHOULD support hyperlinks [S9.6].

16.3 Selection

Viewing consumers that support interactivity MAY support selection and copying [O9.9]. If selection is supported, consumers SHOULD provide a visual cue over or around selected elements [S9.12]. Selection order within an XPS Document SHOULD follow reading order [S9.13].

Consumers MAY use the **FragmentType** attribute of the <StoryFragment> element to determine selection behavior, such as disallowing selection of both the page header and the page contents while allowing independent selection within those stories [O9.10].

16.4 Accessibility

Accessibility refers to features that are important to provide equal access to XPS Documents for users of all abilities. One common example of an accessibility application is a screen reader, which reads the contents of a document aloud for vision-impaired individuals.

16.4.1 Reading Order

In the absence of document structure information provided in the XPS Document, consumers MAY infer the reading order from the position of elements on the page [O9.11], but SHOULD, at minimum, rely on the markup order to determine reading order [S9.14]. Producers SHOULD order the markup in FixedPage parts to reflect the order in which it is intended to be read [S9.15]. When document structure information is present, consumers SHOULD rely on the order of appearance of named elements in the content structure markup to determine reading order [S9.16].

The RECOMMENDED reading order of a page-centric application is as follows [S9.17]:

- Order the content by page.
- Within a page, order by story fragment in the order the <StoryFragment> elements are specified in the StoryFragments part for that page. Producers SHOULD order <StoryFragment> elements in their intended reading order [S9.18].
- Within a <StoryFragment> element, order by <NamedElement> reference.
- Append all un-referenced elements that appear in the fixed page markup, ordered by markup order.

Although producers SHOULD reference every element of the fixed page markup in the content structure markup [S9.10], consumers MUST expose every element of the fixed page markup to an accessibility interface in the determined reading order, even if the elements are not referenced in the content structure markup [M9.9].

Consumers MAY use the **FragmentType** attribute of the <StoryFragment> element to determine reading order by interpreting elements that have **FragmentType** values of Header and Footer as belonging first or last in the reading order, respectively [O9.12].

The RECOMMENDED reading order of a story-centric application is as follows [S9.19]:

- Order content by story in the sequence the <Story> elements appear in the DocumentStructure part. Producers SHOULD order <Story> elements in their intended reading order [S9.20].
- Within a story, order <StoryFragmentReference> elements in the sequence they appear in the DocumentStructure part. Producers SHOULD order <StoryFragmentReference> elements in their intended reading order [S9.21].
- Within a story fragment, order by <NamedElement> references in the StoryFragments part markup.
- Append all un-referenced elements that appear in the fixed page markup, ordered by page number, then markup order.

16.4.2 Screen Reader Applications

Screen reader applications read the contents of the document aloud. A screen reader consumer SHOULD read the document according to its reading order [S9.22]. The application SHOULD use the **UnicodeString** attribute of each <Glyphs> element [S9.23]. In addition, screen readers MAY inspect the **Indices** attribute to resolve potential ambiguities [O9.13].

If the screen reader provides features to navigate the document by structural elements, such as paragraphs or table rows, it SHOULD use any document structure information included in the XPS Document [S9.24].

If the screen reader provides features to describe images, it SHOULD read the text provided in the **AutomationProperties.Name** and **AutomationProperties.HelpText** attributes [S9.25].

If the screen reader provides features to describe hyperlink addresses, it SHOULD read the text provided in the **FixedPage.NavigateUri** attribute [S9.26].

16.4.3 Text Alternatives for Graphics and Images

Images and graphics SHOULD specify text alternatives for images and graphics to make this content accessible to vision-impaired individuals [S9.27]. There are short and long textual descriptions, specified in the **AutomationProperties.Name** and **AutomationProperties.HelpText** attributes of <Path> and <Canvas>, respectively.

The **AutomationProperties.Name** attribute SHOULD contain a short description of the basic contents of the image or vector graphic [S9.27]. [*Example*: "A sitting dog." *end example*]The **AutomationProperties.HelpText** attribute can contain a more detailed description of the image or graphic. [*Example*: "A cocker spaniel with brown eyes, golden fur, and its tongue hanging out. It is sitting on a beanbag directly facing the camera." *end example*]

An image SHOULD specify the **AutomationProperties.Name** and **AutomationProperties.HelpText** attributes on the <Path> element that is filled with an <ImageBrush> [S9.28]. These attributes describe the content specified by the ImageSource attribute of the <ImageBrush> element.

A vector graphic (a collection of one or more <Path> elements representing a single drawing) SHOULD specify the **AutomationProperties.Name** and **AutomationProperties.HelpText** attributes only once, directly on a <Canvas> element wrapping the <Path> elements comprising the graphic [S9.29].

Individual <Path> elements that do not provide any semantic meaning (such as a line between sections or outlining a table) SHOULD NOT specify these text alternative attributes [S9.27].

17. XPS Document Package Features

The XPS Document format extends package-level interleaving and digital signatures as described in the OPC specification.

17.1 Interleaving Optimizations

Interleaving concerns the physical organization of XPS Documents, rather than their logical structure. It allows consumers to linearly process the bytes that make up a physical package from start to finish, without regard for context. In other words, consumers can make correct determinations about the types of logical parts and the presence of relationships on a logical part when consuming packages in a linear fashion. Consumers are never required to return to previously encountered parts and revise their determination of the content type or presence of relationships.

Interleaving is OPTIONAL [O10.1]. However, if the XPS Document is interleaved, these rules SHOULD be followed:

- The Content Types stream SHOULD be interleaved according to the recommendations in the OPC specification [S10.1].
- PrintTicket parts SHOULD be written to the package before the part to which they are attached [S10.2].
- The portion of the relationship data attaching the PrintTicket to a part SHOULD be written to the package before the part to which it is attached or in close proximity to the part to which it is attached [S10.3].
- If no PrintTicket settings are specified for a FixedDocumentSequence, FixedDocument, or FixedPage part, an empty PrintTicket part SHOULD be attached to the part, and the portion of the relationship data attaching the empty PrintTicket SHOULD be written to the package before the part to which it is attached or in close proximity to the part to which it is attached [S10.4].
- The last piece of the Relationships part for a FixedPage part SHOULD be written to the package in close proximity to the first piece of the FixedPage part [S10.5].

Following these recommendations allows more efficient processing by certain consumers. Not following these recommendations could result in less efficient processing by most consumers because they will need to wait until all parts required to process a part (attached PrintTicket, required resources) have been consumed. However, consumers MUST be prepared to correctly process packages in which the PrintTicket or the portion of the relationship data attaching the PrintTicket appears in the package after the affected part [M10.1].

Consumers can choose to parse an XPS Document in a head-first or tail-first manner. Tail-first parsing reveals certain package errors earlier, such as inconsistencies between the ZIP central directory and local file headers. Head-first XPS Document consumers SHOULD attempt to detect inconsistent packages as soon as possible and SHOULD generate an error message, even if they have already processed the pages that resulted in the error [S10.18]. Head-first consumers that discard parts would need to retain the name and length of any discarded part to comply with this recommendation.

17.1.1 Empty PrintTicket

An empty PrintTicket has the following form:

```
<psf:PrintTicket
xmlns:psf="http://schemas.microsoft.com/windows/2003/08/printing/printsch
emaframework" 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 may 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*]

Example 17–2. Optimized interleaving for a multi-threaded parsing architecture

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

Part/Piece	**Markup**
_rels/.rels/[0].piece	`<Relationships xmlns="http://schemas.microsoft.com/package/2005/06/relationships">`
	`<Relationship Type="StartPart" Target="Sequence1"/>`
Sequence1/[0].piece	`<FixedDocumentSequence xmlns="http://schemas.microsoft.com/xps/2005/06">`
	`<DocumentReference Source="FixedDocument1"/>`
FixedDocument1/[0].piece	`<FixedDocument xmlns="http://schemas.microsoft.com/xps/2005/06">`
	`<PageContent Source="Page1"/>`
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>`
Page1	`<FixedPage ="http://schemas.microsoft.com/xps/2005/06" ...>`
	`<Glyphs FontURI="Font1.ttf"/>`
	`</FixedPage>`
Font1.ttf	`...binary font data...`
Other resources	`...resource data...`
FixedDocument1/[1].last.piece	`<PageContent Source="Page2"/>`
Page2	`<FixedPage xmlns="http://schemas.microsoft.com/xps/2005/06" ...>...</FixedPage>`
FixedDocument1/[2].last.piece	`</FixedDocument>`
Sequence1/[1].last.piece	`<DocumentReference Source="FixedDocument2" />`
	`</FixedDocumentSequence>`
FixedDocument2	`<FixedDocument xmlns="http://schemas.microsoft.com/xps/2005/06">`
	`<PageContent Source="Page3"/>`
	`</FixedDocument>`
Page3	`<FixedPage xmlns="http://schemas.microsoft.com/xps/2005/06"`

	`...>...</FixedPage>`
_rels/.rels/[1].last.piece	`</Relationships>`

end example]

17.1.3 Consuming Interleaved Packages

Consumers MUST be able to consume packages regardless of their interleaving structure [M10.2]. Consumers that lack the resources to process a part MUST indicate an error condition [M10.3]. Such a resource constraint exists when a consumer lacks sufficient memory resources to hold enough of the package to resolve all the references required to process a part.

To address resource constraints:

- Consumers MAY discard FixedPage parts once they have been processed [O10.3]
- Consumers MAY discard FixedDocument and FixedDocumentSequence parts after all their child elements and their closing tags have been processed [O10.4].
- In the absence of explicit directives to the contrary (see §17.1.4, "Consumers with Resource Constraints," below), consumers MAY discard parts as directed by the DiscardControl part [O10.5]. Consumers MUST NOT discard any other parts [*Example*: Such as parts containing fonts, images, or other resources *end example*] unless they have the ability to access the parts again [M10.4].

If a consumer encounters a reference to an unknown part, it MUST continue to receive further bytes of the package until the unknown part has been transmitted *or* until the end of the package is reached (indicating an error condition) [M10.5].

17.1.4 Consumers with Resource Constraints

To produce an XPS Document for streaming consumption by consumers with limited memory resources, some producers MAY choose a suitable interleaving order by modeling the resource management behavior of the consumer [O10.6]. These producers, referred to as *drivers*, must have specific knowledge of the XPS Document consumer. Due to resource constraints, some consumers are unable to consume arbitrary XPS Documents and always require assistance from an external driver.

When some consumers with limited memory resources receive a XPS Document in a streaming fashion, there might be an opportunity to discard parts when necessary and reload them again when needed. Producers, such as drivers, that target such consumers SHOULD follow these steps [S10.7]:

- Conservatively model the memory usage of the device.
- Interleave pieces of parts in the correct order.
- Decide when certain parts can be discarded by the consumer and inform the consumer within the package stream (see §17.1.4.1, "DiscardControl Part," below).
- Add to the package a uniquely named copy of a resource that could have been discarded, if the resource is referenced by a part sent later in the stream.

Those later references are also updated to refer to the new copy of the resource.

17.1.4.1 DiscardControl Part

In addition to optimally ordering interleaved parts, producers may support consumers with resource constraints by means of the DiscardControl part. The DiscardControl part is a well-known part containing a list of resources that are safe for the consumer to discard. DiscardControl parts are stored in XPS Documents in an interleaved fashion, allowing a resource-constrained consumer to discard a part as soon as it appears in the DiscardControl part. DiscardControl parts are targeted with a DiscardControl package relationship, as specified in H, “Standard Namespaces and Content Types.” There MUST NOT be more than one DiscardControl package relationship [M10.23]. The DiscardControl part MUST NOT reference itself [M10.6]; doing so is considered an error.

DiscardControl parts that are not well-formed SHOULD NOT be processed and an error SHOULD NOT be reported [S10.8]. The consumer MAY decide to ignore the malformed DiscardControl part in its entirety or from the first malformed node onward [O10.7].

In some cases, producers might rewrite the contents of a package so that parts are provided more than once, allowing consumers to discard a part in order to free resources for additional processing. Each instance of a part MUST be stored as a new, uniquely named part in the package [M10.24].

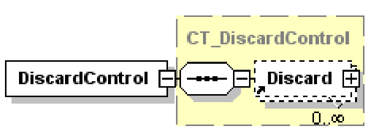
Example 17–3. A DiscardControl part

```
<DiscardControl xmlns="http://schemas.microsoft.com/xps/2005/06/discard-
   control">
   <!-- May discard partname1 as soon as starting to process
      page11.xaml -->
   <Discard SentinelPage="/page11.xaml" Target="/partname1" />
   <!-- May discard partname2 as soon as starting to process
      page13.xaml -->
   <Discard SentinelPage="/page13.xaml" Target="/partname2" />
   ...
</DiscardControl>
```

end example]

17.1.4.1.1 <DiscardControl> Element

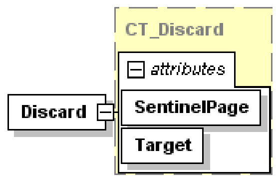
element **DiscardControl**

diagram	
annotation	Contains a list of resources that are safe for a consumer to discard.

17.1.4.1.2 <Discard> Element

element **Discard**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
SentinelPage	xs:anyURI	required			The first fixed page that no longer needs the identified resource in order to be processed.
Target	xs:anyURI	required			The resource that can be safely discarded.

annotation: Identifies a resource that can be safely discarded by a resource-constrained consumer.

Parts that can be discarded are identified in a <Discard> element by the **Target** attribute value, which is expressed as relative to the package root, and by the **SentinelPage** attribute value, which identifies the first FixedPage part that no longer requires the discarded part. (The processing order for FixedPage parts is implied by the order of <PageContent> element references in the FixedDocument part. Therefore, the value of the **SentinelPage** attribute is unambiguous.)

If either the **Target** attribute or the **SentinelPage** attribute contain an invalid reference (refer outside the package), the respective <Discard> element MUST be ignored [M10.7]. If a <Discard> element is encountered where either or both of the **Target** attribute and **SentinelPage** attribute identify a part which has not been processed yet (is still unknown), the <Discard> element SHOULD be retained until both parts identified by the **Target** attribute and **SentinelPage** attribute have been processed or until the end of the package is reached [S10.9].

17.1.5 Interleaving Optimizations and Digital Signatures

In general, it is not feasible to produce well-ordered, interleaved ZIP packages *and* apply digital signatures in a way that enables reasonable consumption scenarios for the following reasons:

- The digital signature parts must be known to consumers before they process other signed parts because the selected hash-methods and transforms must be known. A streaming consumer might not be able to access part data after it has been processed for printing.
- Producers cannot create the digital signature parts before producing the signed packages.
- There are cyclic dependencies with signed relationship parts containing the relationship to the signature parts themselves.

Therefore, when adding a digital signature to an interleaved package, producers of digitally signed documents that are intended for streaming consumption SHOULD add all digital signature parts and the package relationship to the digital signature parts at the beginning of the package, before adding any other part [S10.10].

17.2 Digital Signatures

The digital signature specification for XPS Documents is described in the OPC specification. It allows users to sign arbitrary parts, relationship parts, and individual relationships. Although XPS Documents also use these digital signature mechanisms, they have a specific signature policy and a specific signing mechanism for documents containing co-signing requests.

17.2.1 Signature Policy

This specification defines the signature policy that governs the methods of signing and verifying signatures for XPS Documents. The XPS Document signature policy includes a specific set of signing rules and validity rules. All producers and consumers signing and verifying signatures for end users or applications MUST adhere to these rules consistently [M10.8] to ensure that end users can rely on applications to display accurate signature information.

When signing a document, users may choose to make any of the following actions invalidate the signature:

- Editing core properties
- Adding signatures

Consumers SHOULD inform the end user if an action they are going to take will invalidate an existing signature [S10.11]. Consumers MUST NOT disable functionality based on whether or not an action will invalidate a signature [M10.9].

17.2.1.1 Signing Rules

An XPS Document MUST be considered signed according to the XPS Document signing policy, regardless of the validity of that signature, if the following *signing rules* are followed [M10.10]:

1. The following parts MUST be signed [M10.10]:
    a. The <SignedInfo> portion of the Digital Signature XML Signature part containing this signature.
    b. The FixedDocumentSequence part that is the target of the Start Part package relationship.
    c. All FixedDocument parts referenced in the markup of the FixedDocumentSequence part. (Adding a FixedDocument part to a signed XPS Document will invalidate the signature.)
    d. All FixedPage parts referenced by all signed FixedDocument parts.
    e. All parts associated with each signed FixedPage part by means of a Required Resource relationship (such as fonts, images, color profiles, remote resource dictionaries).
    f. All DocumentStructure parts associated via a Document Structure relationship with all signed FixedDocument parts.

   g. All StoryFragments parts associated via Story Fragments relationship with all signed FixedPage parts.
   h. All SignatureDefinitions parts associated via a Signature Definitions relationship with any signed FixedDocument part. (Once a document is signed, adding any new signature definitions will invalidate the signature.)
   i. All Thumbnail parts associated via a Thumbnail relationship from the package root or with any signed FixedPage or FixedDocument part.
2. The following parts MAY be signed [M10.10]:
   a. The CoreProperties part.
   b. The Digital Signature Origin part.
   c. A Digital Signature Certificate part.
   d. PrintTicket parts.
   e. DiscardControl parts.
3. All relationships with the following RelationshipTypes (see H, "Standard Namespaces and Content Types") MUST be signed [M10.10]:
   a. StartPart relationship from the package root
   b. DocumentStructure relationship from a FixedDocument part
   c. StoryFragments relationship from a FixedPage part
   d. Digital Signature Definitions relationship from a FixedDocument part
   e. Required Resource relationship from a FixedPage part
   f. Restricted Font relationship from a FixedDocument part
   g. Thumbnail relationship from a FixedPage part, a FixedDocument part, or the package root
4. All relationships with the following RelationshipTypes MUST be signed if their Target part is signed [M10.10]:
   a. Core Properties relationship
   b. Digital Signature Origin relationship
   c. Digital Signature Certificate relationship from a Digital Signature XML Signature part
   d. PrintTicket relationship
   e. DiscardControl relationship
5. Relationships with the following RelationshipTypes MAY be signed as a group (they MUST NOT be signed individually) [M10.10]:
   a. All Digital Signature XML Signature relationships from the Digital Signature Origin part (signing all relationships of this RelationshipType will cause this signature to break when a new signature is added).
6. All of the above-referenced parts and relationships MUST be signed using a single digital signature [M10.10].

An XPS Document MUST NOT be considered signed according to the XPS Document signing policy if [M10.11]:

1. Any part not covered by the signing rules above is included in the signature.
2. Any relationship not covered by the signing rules above is included in the signature.

An XPS Document digital signer MUST NOT sign an XPS Document that contains content (parts or relationships parts) to be signed that defines the Markup Compatibility namespace but the signer does not fully understand all elements, attributes, and alternate content representations introduced through the markup compatibility mechanisms [M10.12]. An XPS Document digital signer MAY choose not to sign any content (parts or relationships parts) that defines the Markup Compatibility namespace, even if the content is fully understood [O10.8].

17.2.1.2 Signing Validity

An XPS Document digital signature MUST be shown as an *incompliant digital signature* if [M10.13]:

- It violates any of the signing rules described above regarding parts or relationships that MUST or MUST NOT be signed.

An XPS Document digital signature MUST be shown as a *broken digital signature* if [M10.14]:

- It is not an incompliant digital signature, but the signature fails the signature validation routines described in the Open Packaging Conventions.

An XPS Document digital signature MUST be shown as a *questionable digital signature* if any of the following are true [M10.15]:

- It is not an incompliant or broken digital signature, but the certificate cannot be authenticated against the certificate authority.
- It is not an incompliant or broken digital signature, but the signed content (parts and relationships) contain elements or attributes from an unknown namespace introduced through the Markup Compatibility mechanisms.

An XPS Document digital signature MAY be shown as a questionable digital signature if [O10.9]:

- It is not an incompliant or broken digital signature, but contains some other detectable problem at the discretion of the consumer.

An XPS Document digital signature MUST be shown as a *valid digital signature* if [M10.16]:

- It is not an incompliant, broken, or questionable digital signature.

17.2.1.3 Adding Signatures

XPS Documents MAY be signed more than once [O10.10]. A user who signs an XPS Document might or might not want to allow any additional signing of the document. To prohibit additional signatures in an XPS Document, the signing application MUST sign all the Digital Signature Origin part's relationships of relationship type Digital Signature with the same signature as the rest of the content [M10.17].

17.2.1.4 Certificate Store

XPS Document signatures MUST NOT refer to a remote certificate store (certificate not contained in the XPS Document). All certificates MUST be stored in the XPS Document either as a Certificate part or in the Digital Signature XML Signature part [M10.18].

17.2.1.5 Printing Signed Documents

When printing signed documents, the PrintTicket setting JobDigitalSignatureProcessing SHOULD be used to control the digital signature processing behavior [S10.12]. Producers MAY include the JobDigitalSignatureProcessing setting in the job-level PrintTicket within the XPS Document content [O10.11]. Consumers SHOULD process this PrintTicket setting, if present [S10.12]. For more information, see the Print Schema specification.

Table 17–1. JobDigitalSignatureProcessing PrintTicket settings

Name	Description
PrintInvalidSignatures	Print the job regardless of the validity of the digital signatures. Digital signatures can be ignored.
PrintInvalidSignaturesWithErrorReport	Print the job regardless of the validity of the digital signatures. In the event an invalid signature is encountered, an error page should print at the end of the job. Digital signatures cannot be ignored.
PrintOnlyValidSignatures	Print the job only if all digital signatures are valid. Digital signatures cannot be ignored.

17.2.2 Signature Definitions

In some workflow scenarios, documents must be signed as a means of approving their content. [*Example*: Document producers may be required to sign their documents in order to provide proof of authenticity. *end example*] In other cases, reviewers may be required to co-sign content before it can be submitted for publication. These requirements can be fulfilled with a digitally signed XPS Document.

Whereas the XPS package model supports the signing of arbitrary content in a package, an XPS Document signing workflow requires additional features, including the ability to specify co-signature requirements and to include workflow-specific signature information in the document. XPS Document authors and signing parties provide such information in an XML *signature definition*.

Signature definitions are represented by <SignatureDefinition> elements within a single <SignatureDefinitions> element.

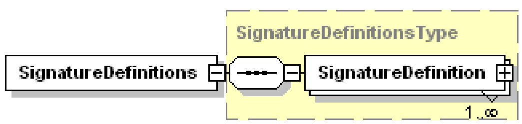
Example 17–4. A SignatureDefinitions part

```
<SignatureDefinitions xmlns="http://schemas.microsoft.com/xps/2005/06/
   signature-definitions">
   <SignatureDefinition SignerName="Dorena Paschke" SpotID="AE2351H">
      <SpotLocation
         PageURI="/Documents/1/Pages/2.fpage"
         StartX="0.0"
         StartY="0.0" />
      <Intent>I have read and agree<Intent>
      <SignBy>2005-08-20</SignBy>
      <SigningLocation>Redmond, WA</SigningLocation>
   <SignatureDefinition>
</SignatureDefinitions>
```

end example]

17.2.2.1 <SignatureDefinitions> Element

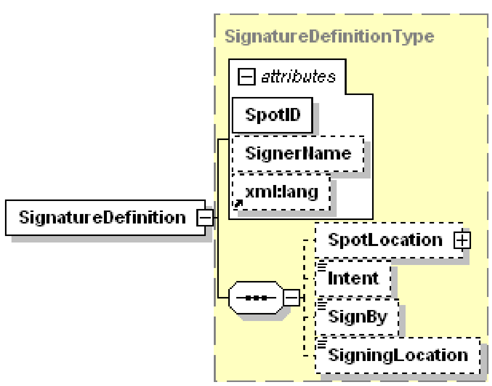
element **SignatureDefinitions**

diagram	
annotation	The root element for the SignatureDefinitions part.

If the SignatureDefinitions part exists, it MUST contain only one <SignatureDefinitions> element [M2.72]. The XML namespace for the <SignatureDefinitions> element is specified in §H.1, "XML Namespace URIs," on page 405.

17.2.2.2 <SignatureDefinition> Element

element **SignatureDefinitionsType/SignatureDefinition**

diagram	

attributes

Name	Type	Use	Default	Fixed	Annotation
SpotID	xs:ID	required			A globally unique identifier for this signature spot.
SignerName	xs:string				A string representing the identity of the individual who is requested to sign the XPS Document, or the name of the individual who has signed the XPS Document.
xml:lang					Specifies the language used for the current element and its descendants. The language is specified according to RFC 3066.

annotation: A single signature definition.

If the SignatureDefinitions part exists, there MUST be *at least* one <SignatureDefinition> element [M2.72].

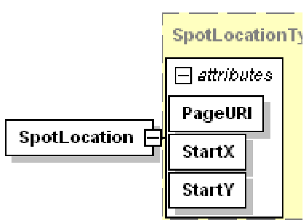
17.2.2.2.1 SpotID Attribute

The SpotID attribute is REQUIRED [M2.72]. This attribute MAY be used to link an existing signature to the <SignatureDefinition> element [O10.12]. The value of this attribute MUST be globally unique to ensure that a Signature part can be linked to only one <SignatureDefinition> element [M2.72]. To link a <SignatureDefinition> to a signature, the value of the **SpotID** MUST be specified in the **Id** attribute of the corresponding <Signature> element in the Digital Signature XML Signature part [M10.19]. For more information, see “Digital Signatures” in the OPC specification.

17.2.2.3 <SpotLocation> Element

element **SignatureDefinitionType/SpotLocation**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
PageURI	xs:anyURI	required			Specifies the page on which the signature spot should be displayed.
StartX	xs:double	required			Specifies the x coordinate of the origin point (upper-left corner) on the page where the

						signature spot should be displayed.
	StartY	xs:double	required			Specifies the y coordinate of the origin point (upper-left corner) on the page where the signature spot should be displayed.
annotation	Specifies where a consumer should place a signature spot.					

The <SpotLocation> element is OPTIONAL [M2.72]. It specifies where an XPS Document viewer should place a visual representation or *signature spot* to indicate that a digital signature has been applied or requested. The viewing consumer SHOULD use the values specified in this element. Due to space and rendering limitations, producers MUST NOT assume that consumers will use these values [M10.20]. If the location specified by this element is not used, it is RECOMMENDED that consumers choose a location that does not contain any page content [S10.13].

The size and shape of the signature spot are determined by the consumer. Consumers MAY choose a size and shape based on the desired display information and page content [O10.13]. However, it is RECOMMENDED that they render signature spots as consistently sized rectangles that include the signer name, the intent, the signing location, and the scope of the XPS Document to be signed [S10.14]. It is also RECOMMENDED that the signature spot be a clickable area used to launch the digital signing process [S10.15].

Figure 17–1. A sample signature spot

17.2.2.4 <Intent> Element

element **SignatureDefinitionType/Intent**

diagram	
annotation	A string that represents the intent to which the signing party agrees when signing the document.

Consumers MUST display the full value of the <Intent> element to the signing party, either in the signature spot or through some other mechanism [M10.21].

[*Note*: Consumers that wish to display signature spots must consider the implications of supporting any Unicode character that can be specified in the <Intent> element. They must also make decisions about the appropriate font face and size to use as well as determine the proper layout and interactivity of the signature spot. These decisions are implementation-specific. *end note*]

17.2.2.5 <SignBy> Element

element **SignatureDefinitionType/SignBy**

diagram	SignBy
annotation	The date and time by which the requested party is to sign the XPS Document.

If specified, the consumer SHOULD NOT allow the signing party to sign the document using this particular signature spot after the date and time specified [S10.16]. The date and time MUST be specified as a complete date plus hours, minutes, and seconds in UTC time, as described in the W3C Note "Date and Time Formats" [M10.22], [*Example*: "2006-12-31T23:59:59Z" or 11:59 PM (UTC) on December 31, 2006. *end example*]

17.2.2.6 <SigningLocation> Element

element **SignatureDefinitionType/SigningLocation**

diagram	SigningLocation
annotation	The legal location where the document is signed.

The <SigningLocation> element MAY be set by the original producer of the XPS Document or by the signing party at the time of signing [O10.14].

17.3 Core Properties

XPS Documents use the Core Properties part described in the OPC. The core properties specified in that part SHOULD refer to the entire fixed payload, including the root FixedDocumentSequence part and the compilation of all FixedDocument parts it references [S10.17].

18. Rendering Rules

The set of rules described here ensures precise and consistent rendering of XPS Document markup across various implementations. Producers MUST generate XPS Documents that can be accurately rendered by following the rules described in this clause [M11.1]. Consumers MUST adhere to the rules described in this clause when rendering XPS Documents [M11.1]. In addition to rules for visual elements, implementation limits are also discussed.

18.1 Coordinate System and Rendering Placement

In the *x,y* coordinate system, one unit is initially equal to 1/96", expressed as a real number. The initial origin of the coordinate system is the top left corner of the fixed page. The *x*-coordinate value increases from left to right; the *y*-coordinate value increases from top to bottom.

A RenderTransform property can be specified on any path, glyphs, or canvas to apply an affine transform to the current coordinate system.

A Transform property can be specified on any visual brush, image brush, linear gradient brush, radial gradient brush, or path geometry to apply an affine transform to the current coordinate system.

18.1.1 Page Dimensions

The logical page dimensions correspond to the media size specified in the application page layout and are specified by the **Width** and **Height** attributes of the <FixedPage> element. Further optional attributes on the <FixedPage> element are used to specify details about the areas of the fixed page that contain rendered content. For more information, see §10.3, "<FixedPage> Element," on page 53.

The physical page dimensions correspond to the media size specified for printing. The physical page dimensions are specified in the PrintTicket PageMediaSize keyword.

The interaction of the logical page dimensions and the physical page dimensions is print-specific. For more information, see §10.3.3, "Page Size Terminology," and §10.3.4, "Media Orientation and Scaling."

18.1.2 Rounding of Coordinates

All computations on coordinate values SHOULD be performed with at least single floating-point precision [S11.1]. Final conversion (after all transforms have been computed) to device coordinates SHOULD retain at least as much fractional precision as a 28.4 fixed-point representation before performing pixel coverage calculations [S11.1].

Very high resolution devices MAY use lower fractional precision to represent device coordinates [O11.1].

When converting from real-number coordinate values to device coordinate values, rounding is performed according to the following rule:

```
coord_D = ROUND(coord_R*16.0)/16
```

Where $\mathtt{coord}_R$ expresses a real-number coordinate value and $\mathtt{coord}_D$ expresses a device coordinate value.

18.1.3 Transforms

XPS Document markup supports affine transforms as expressed through the RenderTransform and Transform properties. An affine transform is represented as a list of six real numbers: m11, m12, m21, m22, dx, dy. (For markup details, see §14.4, "Positioning Content," on page 185.)

The full matrix is as follows:

$$\begin{bmatrix} \mathrm{m11} & \mathrm{m12} & 0 \\ \mathrm{m21} & \mathrm{m22} & 0 \\ \mathrm{dx} & \mathrm{dy} & 1 \end{bmatrix}$$

A given *x,y* coordinate is transformed with a render transform to yield the resulting coordinate x′,y′ by applying the following computations:

```
x' = x * m11 + y * m21 + dx
y' = x * m12 + y * m22 + dy
```

When rendering a child or descendant element, the effective transform used for rendering is the concatenation of all the transforms specified by the RenderTransform or Transform property on parent or ancestor elements, starting from the outermost ancestor.

Non-invertible effective transforms can be specified in markup or occur as a result of limited numerical precision during concatenation. If a non-invertible transform is encountered during rendering, consumers MUST omit rendering the affected element and all of its child and descendant elements [M11.2].

If a non-invertible transform is encountered on a brush (as specified directly on the brush, as a result of the **Viewbox** or **Viewport** attributes, or through concatenation), the brush is treated according to §18.7.1, "Small Tiles," on page 299.

The Width and Height values specified in the **Viewbox** and **Viewport** attributes of an <ImageBrush> or <VisualBrush> element MUST NOT be negative [M2.72].

If a non-invertible transform is encountered on a geometry (as specified directly on the geometry or through concatenation), the geometry MUST be considered to contain no area [M11.3].

A final, device-dependent step using the horizontal resolution and vertical resolution of the device converts the resulting coordinates x′,y′ to device coordinates x″,y″, as follows:

```
x" = x' * R_X/96
y" = y' * R_Y/96
```

Where R_X is the horizontal resolution and R_Y is the vertical resolution of the device, specified in device pixels per inch.

18.1.4 Pixel Center Location, Pixel Placement, and Pixel Inclusion

A pixel covers the range from *x* to *x*+1.

An *ideal* consumer implementation SHOULD render pixels in an 8x8 sub-pixel space, perform an 8x8 box filter sampling, and set the pixel to the resulting color value [S11.2]. Other implementations MAY use different rendering logic as long as it closely approximates this logic [O11.2].

When rendering a shape, a *practical* implementation (such as a bi-tonal printing device) SHOULD turn on each pixel whose center (at *x*+0.5) is covered by the shape, or is touched by the shape with the shape extending beyond the pixel center in the positive *x* or *y* direction of the device [S11.3]. Devices MAY use sub-pixel masking instead [O11.3].

By definition, a shape with an area width of 0 (that is, no included area) does not touch or cover any pixel centers. A stroke with a width of 0 is treated in the same manner.

As a result of these rules, the behavior for very thin lines is implementation-specific:

- An implementation capable of anti-aliasing MAY draw a thin line in a way that blends with the background to varying degrees [O11.4].
- A bi-tonal implementation on a printer MAY draw thin lines with or without drop-outs, or by applying half-toning, depending on the desired output quality [O11.5].

[*Note*: Also see section 11.6.12, “Consistent Nominal Stroke Width” for discussion of thin strokes. *end note*]

18.1.5 Maximum Placement Error

When rendering geometries, consumers SHOULD render curves so they appear smooth from a normal viewing distance [S11.4]. Producers MUST NOT assume a specific placement error for curve decomposition or rely on side-effects of a specific consumer implementation [M11.4].

18.1.6 Pixel Placement for Glyphs

Regardless of other rules expressed here, consumers MAY apply pixel placement rules optimized for character rendering to individual glyphs in a <Glyphs> element [O11.6]. Such rules may result from font hinting applied by the typeface scaler used by a consumer implementation.

18.1.7 Abutment of Shapes

When no anti-aliasing is used, abutting shapes that share the same device coordinates for the end-points and control-points of an edge SHOULD be rendered without overlap and without gaps [S11.5]. Ideally, an implementation SHOULD also follow this rule for shapes that are mathematically abutting without sharing device coordinates for end-points and control-points of edges [S11.5].

18.1.8 Clipping Behavior

Clipping occurs as if a mask were created from the clip geometry according to the pixel placement rules defined in §18.1.4, "Pixel Center Location, Pixel Placement, and Pixel Inclusion," on page 269. An ideal consumer SHOULD create such a mask in an 8x8 sub-pixel space and subsequently draw only those sub-pixels of a shape that correspond to "ON" sub-pixels in the mask [S11.6].

A practical implementation (such as a bi-tonal printing device) SHOULD create a pixel mask according to Point 2 of §18.1.4, "Pixel Center Location, Pixel Placement, and Pixel Inclusion," and subsequently draw only those pixels of a shape that correspond to "ON" pixels in the mask. In creating the mask and drawing the shape, the abutment of shapes rule SHOULD be observed so that no pixel of the shape is drawn that would not have been drawn if the clip geometry were another abutting shape [S11.7]. Devices MAY use sub-pixel masking instead [O11.3].

18.2 Implementation Limits

XPS Document markup does not assume fixed implementation limits. However, consumers may have specific implementation limits imposed by their operating environment. XPS Document markup has been designed so that even complex pages can be represented accurately and with high fidelity.

A typical consumer implementation SHOULD be able to process markup with the characteristics indicated in Table 18–1 [S11.8]. Encountering markup with characteristics outside of the consumer-specific implementation limits MUST cause an error condition [M11.5].

Table 18–1 provides the RECOMMENDED minimum requirements for individual elements [S11.8]. Consumers also have limits on the total number of elements, as imposed by available memory. Producers SHOULD produce only XPS Documents that stay within these implementation limits [S11.8].

In order to process pages that contain a large number of elements, consumers MAY implement support for the DiscardControl part in order to discard elements that have already been processed [O10.5].

Table 18–1. Recommended minimum processing requirements

Characteristic	Type	Limit	Description
Coordinates/transformation matrix elements	Real number	+/− 10^{12}	Largest and smallest coordinate values. Calculations involving numbers close to this limit within a few orders of magnitude are likely to be inaccurate.
Smallest representable non-zero value		+/− 10^{-12}	Coordinate values closest to 0 without rounding to 0. Calculations involving numbers close to this limit within a few orders of magnitude are likely to be inaccurate.
Required precision for		Single	Coordinates are real numbers

Characteristic	Type	Limit	Description
coordinates		floating point	and SHOULD be computed with at least single floating point precision [S11.9].
Nested <Canvas> elements		At least 16	Depth of nested <Canvas> elements.
Nested <VisualBrush> elements		At least 16	Depth of nested <VisualBrush> elements within the Visual property. If the nesting level is higher than the limit, a consumer SHOULD attempt to flatten the nested content to a bitmap representation rather than failing to draw [S11.10].
Total number of points in a path figure		100,000	
Total number of points in a segment		100,000	
Total number of points in a geometry		100,000	
Total number of elements per page		1,000,000	
Number of glyphs per <Glyphs> element		5,000	
Number of elements in a single resource dictionary		10,000	
Total number of elements in all resource dictionaries of an individual page		10,000	
Total number of resource dictionaries in nested canvas scope		Number of nested <Canvas> elements + 1	The <FixedPage> element and each nested <Canvas> element can have at most one associated <ResourceDictionary> element
Number of gradient stops in a gradient brush		100	
Number of fixed documents in a fixed document sequence		1,000	
Number of fixed pages in a fixed document		1,000,000	
Number of dash-gap segments in StrokeDashArray property		No preset limit	Practical number of dash-gap segments depends on the StrokeThickness and the total length of the stroked path.

Characteristic	Type	Limit	Description
Total size of XPS Document markup per page	Bytes	64,000,000	Total size of markup after removing all unnecessary whitespace (according to the schema in B), assuming markup elements are specified in the default namespace without namespace prefixes, and assuming the most compact representation of all attributes using abbreviated syntax where possible.

18.3 Gradient Computations

To ensure the greatest possible consistency among consumers, gradients SHOULD be rendered according to the guidelines described in this section [S11.11].

18.3.1 All Gradients

Linear gradients and radial gradients share a common set of recommended operations for pre-processing gradient stops and blending colors. These are described below.

18.3.1.1 Gradient Stop Pre-Processing

Consumers SHOULD pre-process gradient stops for all gradients using the following steps [S11.12]:

1. Sort all gradient stops by their respective offset values in ascending order. When two or more gradient stops have the same offset value, preserve their relative order from the markup while sorting. When more than two gradient stops have the same offset value, remove all but the first and last gradient stops having the same offset value.
2. If no gradient stop with an offset of 0.0 exists,
    a. And no gradient stop with an offset less than 0.0 exists, create an artificial gradient stop having an offset of 0.0 and a color of the gradient stop with the smallest offset value.
    b. And a gradient stop with an offset less than 0.0 exists and a gradient stop with an offset greater than 0.0 exists, create an artificial gradient stop having an offset of 0.0 and a color interpolated between the two gradient stops surrounding 0.0. Discard all gradient stops with an offset less than 0.0.
    c. And a gradient stop with an offset less than 0.0 exists and no gradient stop with an offset greater than 0.0 exists, create an artificial gradient stop having an offset of 0.0 and a color of the gradient stop with the largest offset value. Discard all gradient stop elements with an offset less than 0.0.
3. If no gradient stop with an offset of 1.0 exists,

a. And no gradient stop with an offset of greater than 1.0 exists, create an artificial gradient stop having an offset of 1.0 and a color equal to the color of the gradient stop with the largest offset value.

b. And a gradient stop with an offset greater than 1.0 exists and a gradient stop with an offset less than 1.0 exists, create an artificial gradient stop having an offset of 1.0 and a color interpolated between the two surrounding gradient stops. Discard all gradient stops with an offset greater than 1.0.

c. And a gradient stop with an offset greater than 1.0 exists and no gradient stop with an offset less than 1.0 exists, create an artificial gradient stop having an offset of 1.0 and a color equal to that of the gradient stop with the smallest offset value. Discard all gradient stops with an offset greater than 1.0.

18.3.1.2 Blending Colors

If any gradient stops use an sRGB or scRGB color specification or the consumer does not understand the PageBlendColorSpace PrintTicket setting, consumers SHOULD blend colors between gradient stops in the color space indicated by the **ColorInterpolationMode** attribute of the gradient brush [S11.13]. If none of the gradient stop elements uses an sRGB or scRGB color specification and the consumer understands the PageBlendColorSpace PrintTicket setting, the PageBlendColorSpace PrintTicket setting SHOULD be used [S11.13].

The function used for blending is:

```
BLEND(offset, clo, chi)
```

Where the offset is between 0 and 1. c_{lo} and c_{hi} designate the color values for an offset of 0 and 1, respectively.

If a **ColorInterpolationMode** value of SRgbLinearInterpolation is used, the BLEND() function SHOULD convert the color values to sRGB first, and then perform a linear interpolation between them [S11.14].

If a **ColorInterpolationMode** value of ScRgbLinearInterpolation is used, the BLEND() function SHOULD convert the color values to scRGB first, and then perform a linear interpolation between them [S11.15].

If blending is performed in the color space identified by the PageBlendColorSpace PrintTicket setting, it SHOULD be a linear, channel-by-channel blend operation [S11.13].

In the presence of transformations or when individual gradient stops are very close (separated by a few pixels or less in the device space), the local color gradient at the offset used in the BLEND() function may be large, resulting in a large change over the extent of a single device pixel. In this case, it is RECOMMENDED that the BLEND() function interpolate the gradient over the extent of each device pixel [S11.16]. However, the behavior MAY differ from this recommendation in an implementation-specific manner [O11.7] and, therefore, producers SHOULD NOT rely on a specific effect for such dense gradient specifications [S11.16].

As a consequence of this interpolation, radial gradients that define the gradient origin on or outside ellipse create an “outside” area that can be rendered inconsistently. The radial gradients that are affected are those that define multiple gradient stops that are of different colors and are very close in Offset value to 0.0 or 1.0 (the gradient end points), for radial gradients with a **SpreadMethod** value of Repeat or Reflect, respectively. For these affected gradients, consumers MAY use an interpolated color value for the outside area [O11.8]. Depending on the resolution, this can result in different colors than those defined by the gradient end points. The closer a gradient stop is to the affected gradient end point, the more the rendering results on different consumers and at different display resolutions can differ. Producers SHOULD therefore either avoid such close gradient stops to the gradient end point when specifying radial gradients where the outside area is visible or avoid specifying radial gradients with a gradient origin on or outside the ellipse (in which case there is no outside area) to ensure consistent rendering results [S11.17].

18.3.2 Linear Gradients

Consumers SHOULD render an element filled with a linear gradient brush such that the appearance is the same as if the following steps had been taken [S11.11]:

1. Transform the **StartPoint** and **EndPoint** attribute values using the current effective render transform (including the render transform for the element being filled by the linear gradient brush and the brush's transform itself).

2. If the **SpreadMethod** value is Pad, the colors of points on the line defined by the **StartPoint** and **EndPoint** attributes are defined by interpolating the coordinates linearly, and each color component (such as R, G, B for sRGB and scRGB) as well as the alpha component is interpolated between the component values of the closest enclosing gradient stops:

```
For each offset (real number) t < 0:
{
   x(t) = (EndPoint_x-StartPoint_x)*t+StartPoint_x
   y(t) = (EndPoint_y-StartPoint_y)*t+StartPoint_y
   c(t) = c_first
   a(t) = a_first
}
```

   Where c is the color component and a is the alpha component. c_{first} are the color component values of the first gradient stop (after sorting) and a_{first} is the alpha component value at the first gradient stop (after sorting).

```
For each offset (real number) 0 <= t <= 1:
{
   x(t) = (EndPoint_x-StartPoint_x)*t+StartPoint_x
   y(t) = (EndPoint_y-StartPoint_y)*t+StartPoint_y
   c(t) = BLEND((t-t_lo)/(t_hi-t_lo),c_lo,c_hi)
   a(t) = [(t-t_lo)/(t_hi-t_lo)]*(a_hi-a_lo)+a_lo
}
```

   Where t_{lo} and t_{hi} are the offsets, c_{lo} and c_{hi} are the color component values at the closest enclosing gradient stops (that is, $t_{lo} <= t <= t_{hi}$) and a_{lo} and a_{hi} are the alpha component values at the closest enclosing gradient stops ($t_{lo} <= t <= t_{hi}$).

```
For each offset (real number) t > 1:
{
   x(t) = (EndPoint_x-StartPoint_x)*t+StartPoint_x
   y(t) = (EndPoint_y-StartPoint_y)*t+StartPoint_y
   c(t) = c_last
   a(t) = a_last
}
```

Where c_{last} are the color component values of the last gradient stop (after sorting) and a_{last} is the alpha component value at the last gradient stop (after sorting).

3. If the **SpreadMethod** value is Repeat, the colors of points on the line defined by the **StartPoint** and **EndPoint** attributes are defined by interpolating the coordinates linearly, and each color component (such as R, G, B for sRGB and scRGB) as well as the alpha component is interpolated between the component values of the closest enclosing gradient stops:

```
For each repetition (all integers) N:
{
   For each offset (real number) 0 <= t < 1:
   {
      x(t) = (EndPoint_x-StartPoint_x)*(N+t)+StartPoint_x
      y(t) = (EndPoint_y-StartPoint_y)*(N+t)+StartPoint_y
      c(t) = BLEND((t-t_lo)/(t_hi-t_lo),c_lo,c_hi)
      a(t) = [(t-t_lo)/(t_hi-t_lo)]*(a_hi-a_lo)+a_lo
   }
}
```

Where c is the color component and a is the alpha component. t_{lo} and t_{hi} are the offsets, c_{lo} and c_{hi} are the color component values at the closest enclosing gradient stops (that is, t_{lo} <= t <= t_{hi}) and a_{lo} and a_{hi} are the alpha component values at the closest enclosing gradient stops (t_{lo} <= t <= t_{hi}).

4. If the **SpreadMethod** value is Reflect, the colors of points on the line defined by the **StartPoint** and **EndPoint** attributes are defined by interpolating the coordinates linearly, and each color component (such as R, G, B for sRGB and scRGB) as well as the alpha component is interpolated between the component values of the closest enclosing gradient stops:

```
For each repetition (all integers) N:
{
   For each offset (real number) 0 <= t <= 1:
   {
      If (N is EVEN)
      {
         x(t) = (EndPoint_x-StartPoint_x)*(N+t)+StartPoint_x
         y(t) = (EndPoint_y-StartPoint_y)*(N+t)+StartPoint_y
      }
      Else
      {
         x(t) = (EndPoint_x-StartPoint_x)*(N+1-t)+StartPoint_x
         y(t) = (EndPoint_y-StartPoint_y)*(N+1-t)+StartPoint_y
      }
      c(t) = BLEND((t-t_lo)/(t_hi-t_lo),c_lo,c_hi)
      a(t) = [(t-t_lo)/(t_hi-t_lo)]*(a_hi-a_lo)+a_lo
   }
}
```

Where c is the color component and a is the alpha component. t_{lo} and t_{hi} are the offsets, c_{lo} and c_{hi} are the color component values at the closest enclosing gradient stops (that is, t_{lo} <= t <= t_{hi}) and a_{lo} and a_{hi} are the alpha component values at the closest enclosing gradient stops (t_{lo} <= t <= t_{hi}).

5. The colors of points not on the extended line defined by the **StartPoint** and **EndPoint** attributes are the same as the color of the closest point on the line defined by the **StartPoint** and **EndPoint** attributes, measured in the coordinate space as transformed by the current effective render transform

(including the render transform for the element being filled by the linear gradient brush and the brush's transform itself).

6. Clip the resulting set of points to the intersection of the current clip geometry and the path or glyphs to be filled. Both the clip and path (or glyphs) must be transformed according to the current effective render transform, including the render transform for the element being filled, but *not* including the transform of the linear gradient brush.

For purposes of the above steps, the closest enclosing gradient stops mean the gradient stops that, if the relative sequencing of the gradient stop offsets in the markup order is respected, are numerically closest to the interpolation point if that interpolation point were converted to an offset value and inserted in a sorted fashion into the list of gradient stops. [*Example*: If a gradient contains three gradient stops at offset values 0.0, 0.0, and 1.0, the closest enclosing gradient stops for any value 0 <= value <= 1 are the second gradient stop (offset 0.0) and the third gradient stop (offset 1.0). *end example*]

18.3.3 Radial Gradients

Consumers SHOULD render an element filled with a radial gradient brush such that the appearance is the same as if these steps had been followed [S11.11]:

1. The boundary of the area filled by a radial gradient brush is defined by interpolating ellipses from the **GradientOrigin** value to the circumference of the ellipse centered at the point specified by the **Center** attribute with radii equal to the **RadiusX** and **RadiusY** attribute values(the interpolated ellipses and point being transformed by the current effective render transform, including the render transform for the element being filled by the radial gradient brush and the brush's transform itself). If the gradient origin is outside the circumference of the ellipse specified, the effect will be as if a cone were drawn, tapering to the gradient origin.

2. If the **SpreadMethod** value is Pad, the centers and radii of the interpolated ellipses are defined by linearly interpolating the center of the ellipse from the **GradientOrigin** attribute value to the **Center** attribute value, and simultaneously linearly interpolating the radii of the ellipse from 0 to the **RadiusX** and **RadiusY** attribute values:

```
For each offset (real number) 0 <= t <= 1:
{
   cx(t) = (Centerx-GradientOriginx)*t+GradientOriginx
   cy(t) = (Centery-GradientOriginy)*t+GradientOriginy
   rx(t) = RadiusX*t
   ry(t) = RadiusY*t
}
```

   The ellipses defined by the interpolation are transformed by the current effective render transform, including the render transform for the element being filled by the radial gradient brush and the brush's transform itself.

3. The colors of the points within the boundary of this shape are defined as the color of the smallest interpolated ellipse containing the point. The color of an interpolated ellipse is defined by interpolating each color component (such as R, G, B for sRGB and scRGB) as well as the alpha component between the component values of the closest enclosing gradient stops:

```
For each offset (real number) 0 <= t <= 1:
{
   c(t) = BLEND((t-t_lo)/(t_hi-t_lo),c_lo,c_hi)
   a(t) = [(t-t_lo)/(t_hi-t_lo)]*(a_hi-a_lo)+a_lo
}
```

Where t_{lo} and t_{hi} are the offsets, c_{lo} and c_{hi} are the color component values at the closest enclosing gradient stops (that is, $t_{lo} <= t <= t_{hi}$) and a_{lo} and a_{hi} are the alpha component values at the closest enclosing gradient stops ($t_{lo} <= t <= t_{hi}$).

4. If the **SpreadMethod** value is Repeat, the centers and radii of the interpolated ellipses are defined by linearly interpolating the center of the ellipse from the **GradientOrigin** attribute value to the **Center** attribute value, and simultaneously linearly interpolating the radii of the ellipse from 0 to **RadiusX** and **RadiusY** attribute values:

```
For each repetition (all non-negative integers) N:
{
   For each offset (real number) 0 <= t < 1:
   {
      c_x(t) = (Center_x-GradientOrigin_x)*(N+t)+GradientOrigin_x
      c_y(t) = (Center_y-GradientOrigin_y)*(N+t)+GradientOrigin_y
      r_x(t) = RadiusX*(N + t)
      r_y(t) = RadiusY*(N + t)
   }
}
```

The ellipses defined by the interpolation are transformed by the current effective render transform, including the render transform for the element being filled by the radial gradient brush and the brush’s transform itself.

5. The colors of the points within the boundary of this shape are defined as the color of the smallest interpolated ellipse containing the point. The color of an interpolated ellipse is defined by interpolating each color component (such as R, G, B for sRGB and scRGB) as well as the alpha component between the component values of the closest enclosing gradient stops:

```
For each repetition (all non-negative integers) N:
{
   For each offset (real number) 0 <= t < 1:
   {
      c(t) = BLEND((t-t_lo)/(t_hi-t_lo),c_lo,c_hi)
      a(t) = [(t-t_lo)/(t_hi-t_lo)]*(a_hi-a_lo)+a_lo
   }
}
```

Where t_{lo} and t_{hi} are the offsets, c_{lo} and c_{hi} are the color component values at the closest enclosing gradient stops (that is, $t_{lo} <= t <= t_{hi}$) and a_{lo} and a_{hi} are the alpha component values at the closest enclosing gradient stops ($t_{lo} <= t <= t_{hi}$).

6. If the **SpreadMethod** value is Reflect, the centers and radii of the interpolated ellipses are defined by linearly interpolating the center of the ellipse from the **GradientOrigin** attribute value to the **Center** attribute value, and simultaneously linearly interpolating the radii of the ellipse from 0 to the **RadiusX** and **RadiusY** attribute values:

```
For each non-negative integer N:
{
   For each offset (real number) 0 <= t <= 1:
```

```
   {
      c_x(t) = (Center_x-GradientOrigin_x)*(N+t)+GradientOrigin_x
      c_y(t) = (Center_y-GradientOrigin_y)*(N+t)+GradientOrigin_y
      r_x(t) = RadiusX*(N+t)
      r_y(t) = RadiusY*(N+t)
   }
}
```

The ellipses defined by the interpolation are transformed by the current effective render transform, including the render transform for the element being filled by the radial gradient brush and the brush's transform itself.

7. The colors of the points within the boundary of this shape are defined as the color of the smallest interpolated ellipse containing the point. The color of an interpolated ellipse is defined by interpolating each color component (such as R, G, B for sRGB and scRGB) as well as the alpha component between the component values of the closest enclosing gradient stops:

```
For each non-negative integer N:
{
   For each offset (real number) 0 <= t <= 1:
   {
      If N is ODD
         t' = 1-t
      Else
         t' = t

      c(t) = BLEND((t'-t_lo)/(t_hi-t_lo),c_lo,c_hi)
      a(t) = [(t'-t_lo)/(t_hi-t_lo)]*(a_hi-a_lo)+a_lo
   }
}
```

Where t_{lo} and t_{hi} are the offsets, c_{lo} and c_{hi} are the color component values at the closest enclosing gradient stops (that is, $t_{lo} <= t <= t_{hi}$) and a_{lo} and a_{hi} are the alpha component values at the closest enclosing gradient stops ($t_{lo} <= t <= t_{hi}$).

8. The colors of points outside the boundary of this shape (points which cannot be drawn by any combination of non-negative N and t) are defined as having the color and alpha defined in the gradient stop with the offset of 0.0 for radial gradients with a **SpreadMethod** value of Reflect and the color and alpha defined in the gradient stop with the offset of 1.0 for radial gradients with a **SpreadMethod** value of Repeat or Pad. The colors outside of the boundary of this shape may also vary in an implementation-specific manner (see §18.3.1.2, "Blending Colors" for more details).

9. Clip the resulting set of points by the intersection of the current clip geometry and the path or glyphs to be filled. Both the clip and path (or glyphs) must be transformed according to the current effective render transform, including the render transform for the element being filled, but *not* including the transform of the radial gradient brush.

For purposes of the above steps, the closest enclosing gradient stops mean the gradient stops that, if the relative sequencing of the gradient stop offsets in the markup order is respected, are numerically closest to the interpolation point if that interpolation point were converted to an offset value and inserted in a sorted fashion into the list of gradient stops. [*Example*: If a gradient contains three gradient stops at offset values 0.0, 0.0, and 1.0, the closest enclosing gradient stops for any value

0 <= value <= 1 are the second gradient stop (offset 0.0) and the third gradient stop (offset 1.0). *end example*]

18.4 Opacity Computations

Opacity is used to transparently blend two elements when rendering, also known as alpha blending. The value of the Opacity property ranges from 0.0 (fully transparent) to 1.0 (fully opaque), inclusive. Values outside of this range are invalid.

The opacity is applied through the following computations, assuming source and destination values are not pre-multiplied. All opacity calculations SHOULD be performed with at least 8-bit precision to provide sufficient quality for nested content [S11.18].

Individual pixels are blended as defined below.

Table 18–2. Opacity computation symbols

Symbol	Description
O_E	Opacity attribute of element
O_M	Alpha value at corresponding pixel position in the **OpacityMask** attribute value
A_S	Alpha value present in source color
R_S	Red value present in source color
G_S	Green value present in source color
B_S	Blue value present in source color
A_D	Alpha value already present in destination surface
R_D	Red value already present in destination surface
G_D	Green value already present in destination surface
B_D	Blue value already present in destination surface
A*	Resulting Alpha value for destination surface
R*	Resulting Red value for destination surface
G*	Resulting Green value for destination surface
B*	Resulting Blue value for destination surface

All values designated with a T subscript (as in R_{T1}) are temporary values.

The opacity is calculated as follows:

1. Multiply source alpha value with opacity value and alpha value of opacity mask.

$$A_{S1} = A_S*O_E*O_M$$

2. Pre-multiply source alpha.

   Omit this step if the source data specifies pre-multiplied alpha (see §18.4.1, "Pre-Multiplied Alpha and Superluminous Colors," on page 281 for details).

$$\begin{aligned}A_{T1} &= A_{S1}\\ R_{T1} &= R_S*A_{S1}\\ G_{T1} &= G_S*A_{S1}\end{aligned}$$

$$B_{T1} = B_S*A_{S1}$$

3. Pre-multiply destination alpha.

   Omit this step in consumers supporting superluminous colors (see §18.4.1, “Pre-Multiplied Alpha and Superluminous Colors,” on page 281 for details).

$$\begin{aligned}A_{T2} &= A_D\\ R_{T2} &= R_D*A_D\\ G_{T2} &= G_D*A_D\\ B_{T2} &= B_D*A_D\end{aligned}$$

4. Blend.

   See §18.4.1, “Pre-Multiplied Alpha and Superluminous Colors,” on page 281 for special case handling.

$$\begin{aligned}A_{T3} &= (1-A_{T1})*A_{T2}+A_{T1}\\ R_{T3} &= (1-A_{T1})*R_{T2}+R_{T1}\\ G_{T3} &= (1-A_{T1})*G_{T2}+G_{T1}\\ B_{T3} &= (1-A_{T1})*B_{T2}+B_{T1}\end{aligned}$$

5. Reverse pre-multiplication.

   Omit this step in consumers supporting superluminous colors. See also §18.4.1, “Pre-Multiplied Alpha and Superluminous Colors,” on page 281.

   The resulting color channel values are divided by the resulting alpha value. If the resulting alpha value is 0, all color channels are set to 0 by definition, as expressed in the If condition below. Each of R_{T3}, G_{T3}, B_{T3} is smaller than or equal to A_{T3} and, therefore, each of the resulting R*, G*, B* is in the valid interval of [0.0,1.0] after the pre-multiplication is reversed.

```
If A_T3 = 0
{
   set all A* R* G* B* to 0.
}
Else
{
   A* = A_T3
   R* = R_T3/A_T3
   G* = G_T3/A_T3
   B* = B_T3/A_T3
}
```

When blending colors in a color space other than sRGB, color channels are independently interpolated in a manner analogous to the RGB channel blending method described above.

18.4.1 Pre-Multiplied Alpha and Superluminous Colors

The alpha information in TIFF images using an ExtraSamples tag value of 1 and in Windows Media Photo images using pixel formats WICPixelFormat32bppPBGRA, WICPixelFormat64bppPRGBA or WICPixelFormat128bppPRGBAFloat MUST be interpreted as pre-multiplied alpha information [M11.6]. In certain scenarios (such as when rendering 3D scenes to a bitmap), producers MAY choose to create pre-multiplied bitmap data specifying "superluminous" colors [O11.9].

Superluminous colors are defined as pre-multiplied color values with an alpha value smaller than the individual color channel values but greater than or equal to 0.

The effect of composing superluminous colors on a background is similar to adding additional light of the source color to the destination, as opposed to regular alpha composition which works more like a colored filter. One can easily verify this statement by substituting 0 for A_{T1} in step 4 of the above opacity computations, which is simplified as follows (note that R_{T1} , G_{T1} , B_{T1} are not 0, because the pre-multiplication in step 2 has been skipped):

$$
\begin{aligned}
A_{T3} &= A_{T2} \\
R_{T3} &= R_{T2}+R_{T1} \\
G_{T3} &= G_{T2}+G_{T1} \\
B_{T3} &= B_{T2}+B_{T1}
\end{aligned}
$$

Consumers supporting superluminous colors retain all temporary information in pre-mulitplied formats. Note, that throughout the XPS specification non-pre-multiplied alpha processing is assumed. It is up to the implementer of such a consumer to identify equivalent composition and rendering rules for processing in pre-multiplied space.

Also note, when composing superluminous colors, management of out-of-gamut colors SHOULD be deferred until the result is rendered to the final target, at which point out-of-gamut colors are clipped or color managed [S11.19].

Consumers MAY handle superluminous colors or MAY instead choose to convert pre-multiplied source data containing superluminous colors to non-pre-multiplied data before composition by ignoring the superluminous portion of each color channel value [O11.10], as described in the following steps:

```
For each superluminous pixel with A < R or A < G or A < B
{
   If A = 0
   {
      R* = 1
      G* = 1
      B* = 1
   }
   Else
   {
      A* = A
      R* = min(R/A,1)
      G* = min(G/A,1)
      B* = min(B/A,1)
    }
}
```

18.5 Composition Rules

XPS Document page markup uses the painter's model with alpha channel. Composition MUST have the same effect as the application of the following rules, in sequence [M11.7]:

1. In order to render a fixed page or canvas, a surface is created to hold the drawing content as it is composed. The color and appearance of this surface SHOULD match the destination color and appearance, typically a solid white background for a fixed page or transparent for a canvas [S11.20]. An implementation MAY choose to meet this goal by always initializing this surface's alpha channel to 0.0 (transparent) and the color value to black [O11.5].
2. The fixed page or canvas represents a surface onto which child elements are drawn. The child elements are drawn in the order they appear in markup. In practice, an implementation might represent the surface by a bitmap buffer large enough to hold all the drawing content produced when the child elements are rendered.
3. The contents appearing on the surface of canvas are transformed using the affine transform specified by the RenderTransform property of the canvas. (A fixed page does not have a RenderTransform property.)
4. All child elements are rendered to the surface and clipped to the imageable area of the physical display (such as a sheet of paper) of the fixed page or according to the Clip property of a canvas. The geometry value of the canvas' Clip property is also transformed using the affine transform specified by the RenderTransform property of the canvas.
5. If a path has a Stroke and a Fill property, and also specifies Opacity or OpacityMask property values, additional composition steps must be followed:
    a. Create a temporary canvas with the opacity, opacity mask, clip, and render transform specified by the path.
    b. Create a copy of the original path, remove all but the Fill property from the copy, and add the copy to the temporary canvas.
    c. Create another copy of the original path, remove all but the stroke-related properties (such as Stroke, StrokeThickness, and StrokeDashArray) from the copy, and add the copy to the temporary canvas.
    d. Do not draw the original path.
    e. Draw the temporary canvas, while recursively applying the composition rules.
6. If a grouping element (a <Canvas> element) has an Opacity or OpacityMask property, additional composition steps must be followed:
    a. Create a temporary surface and set its alpha channel to 0.0 (transparent) and its color value to black.
    b. Compose all child elements of the grouping element onto the temporary surface, while recursively applying the composition rules.
    c. Cumulatively apply the opacity of the grouping element and opacity mask to the alpha channel of the temporary surface.

    d. Draw the contents of the temporary surface onto the containing surface.

7. If a non-grouping element (a <Path> or <Glyphs> element) has an Opacity property, an OpacityMask property, or a fill or stroke using transparency, the following additional composition steps must be taken:
    a. If the element has a RenderTransform property, apply it to the element and its Clip, Fill, Stroke, and OpacityMask properties, if present.
    b. Create a mask from the set of all painted pixels representing the child element (after the Clip property of the element has been applied).
8. Combine the Fill or Stroke property with the OpacityMask and the Opacity property and apply to the surface through the computed mask. For more information, see §14.1, "Opacity," on page 172.

The behavior that results from this process is:

- Opacity is not applied cumulatively to self-overlapping areas created when rendering an individual <Glyphs> element.
- Opacity is not applied cumulatively to self-overlapping areas created by <PathFigure> elements within the same path (see Example 18–1 on page 284).
- Opacity is not applied cumulatively if the border of a path has self-intersections. When the border of a path is stroked, the area of the path is filled by first applying the brush specified by the Fill property. After filling the area, the border is drawn using the stroke-related properties including the brush specified by the Stroke property, with half the stroke width extending outside the filled area and half extending inside (see Example 18–2 on page 285). If the path has self-intersections, the opacity is not accumulated.
- The color of the stroke and the color of the filled area are combined on the inside half of a stroked border (overlapping the filled area of the path) if the brush specified by the Stroke property is transparent.
- If a path that has a stroked border has an opacity of less than 1.0 or an opacity mask, the path (filled area and stroked border) is first rendered onto a temporary surface using an opacity of 1.0 and no opacity mask (while preserving any transparency of the fill or the stroked border themselves), and the resulting figure is drawn onto the background using the specified opacity and opacity mask (see Example 18–3 on page 285).

18.5.1 Optimization Guidelines

The composition rules above describe the behavior of an ideal implementation. Practical implementations can optimize the processing of the composition rules according to the following guidelines:

1. If all elements on a canvas and the canvas itself are opaque (an opacity of 1.0) and parent or ancestor <Canvas> elements are also opaque, the elements MAY be drawn directly to the containing fixed page (or canvas), provided all render transform and clip values are observed [O11.12].
2. If an element is fully transparent (an opacity of 0.0), it MAY be skipped [O11.13].

3. If a canvas has an opacity of 0.0, it and all of its child and descendant elements MAY be skipped [O11.14].
4. If a canvas has a Clip property with no contained area, the canvas and all of its child and descendant elements MAY be skipped [O11.15].
5. When creating a temporary surface, a consumer MAY further restrict the size of the temporary surface by the effective extent of the geometry specified by the Clip property of the canvas [O11.16].
6. A consumer MAY use methods to achieve transparency other than creating a temporary surface [O11.17]. Such methods MAY include planar mapping (that is, computation of intersections of transparent elements and resulting colors) [O11.17].

18.5.2 Composition Examples

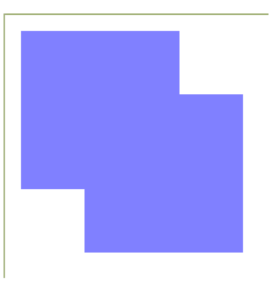
The following examples illustrate the composition rules described above.

Example 18–1. Path opacity behavior for overlapping path figures

In the following markup, opacity is not applied cumulatively to self-overlapping areas created by path figures within the same path.

```
<Path Opacity="0.5">
   <Path.Fill>
      <SolidColorBrush Color="#0000FF" />
   </Path.Fill>
   <Path.Data>
      <PathGeometry FillRule="NonZero">
         <PathFigure StartPoint="10,10" IsClosed="true">
            <PolyLineSegment Points="110,10 110,110 10,110 10,10" />
         </PathFigure>
         <PathFigure StartPoint="50,50" IsClosed="true">
            <PolyLineSegment Points="150,50 150,150 50,150 50,50" />
         </PathFigure>
      </PathGeometry>
   </Path.Data>
</Path>
```

This markup is rendered as follows:



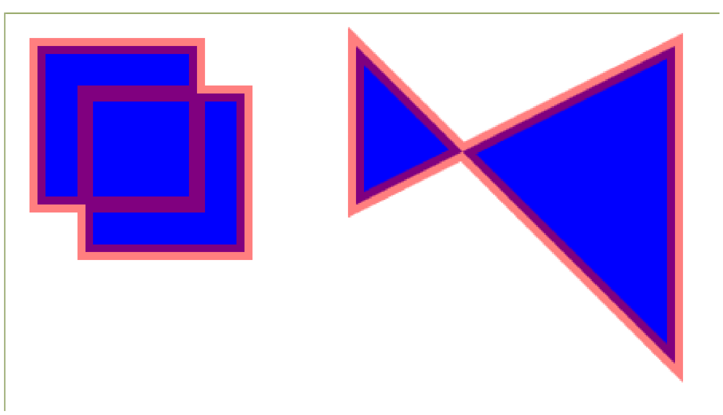
end example]

Example 18–2. Opacity behavior of path stroke intersections

In the following markup, opacity is not applied cumulatively if the border of a path has self-intersections.

```
<Path Stroke="#80FF0000" StrokeThickness="10">
   <Path.Fill>
      <SolidColorBrush Color="#0000FF" />
   </Path.Fill>
   <Path.Data>
      <PathGeometry FillRule="NonZero">
         <PathFigure StartPoint="20,20" IsClosed="true">
            <PolyLineSegment Points="120,20 120,120 20,120 20,20" />
         </PathFigure>
         <PathFigure StartPoint="50,50" IsClosed="true">
            <PolyLineSegment Points="150,50 150,150 50,150 50,50" />
         </PathFigure>
      </PathGeometry>
   </Path.Data>
</Path>
<Path Stroke="#80FF0000" StrokeThickness="10" StrokeMiterLimit="10">
   <Path.Fill>
      <SolidColorBrush Color="#0000FF" />
   </Path.Fill>
   <Path.Data>
      <PathGeometry>
         <PathFigure StartPoint="220,20" IsClosed="true">
            <PolyLineSegment Points="420,220 420,20 220,120" />
         </PathFigure>
      </PathGeometry>
   </Path.Data>
</Path>
```

This markup is rendered as follows:



end example]

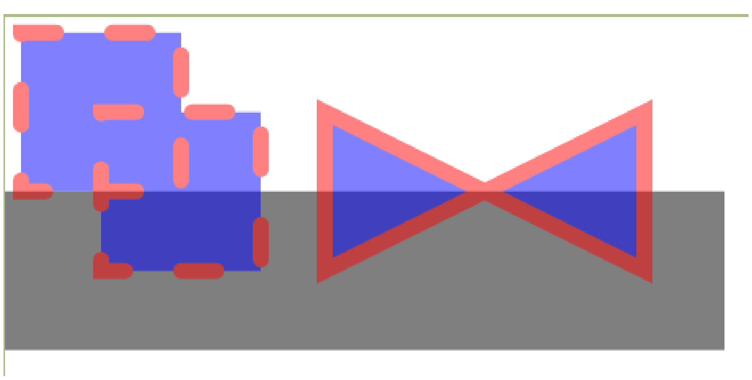
Example 18–3. Opacity behavior of paths with stroked edges

The following markup describes a path with a stroked border and an opacity of less than 1.0:

```
<Path>
   <Path.Fill>
```

```
         <SolidColorBrush Color="#7F7F7F" />
      </Path.Fill>
      <Path.Data>
         <PathGeometry>
            <PathFigure StartPoint="0,110" IsClosed="true">
               <PolyLineSegment Points="450,110 450,210 0,210" />
            </PathFigure>
         </PathGeometry>
      </Path.Data>
   </Path>
   <Path
      Stroke="#FF0000"
      StrokeThickness="10"
      StrokeDashArray="2.2 3.5"
      StrokeDashCap="Round"
      Opacity="0.5">
      <Path.Fill>
         <SolidColorBrush Color="#0000FF" />
      </Path.Fill>
      <Path.Data>
         <PathGeometry FillRule="NonZero">
            <PathFigure StartPoint="10,10" IsClosed="true">
               <PolyLineSegment Points="110,10 110,110 10,110 10,10" />
            </PathFigure>
            <PathFigure StartPoint="60,60" IsClosed="true">
               <PolyLineSegment Points="160,60 160,160 60,160 60,60" />
            </PathFigure>
         </PathGeometry>
      </Path.Data>
   </Path>
   <Path Stroke="#FF0000" StrokeThickness="10" Opacity="0.5">
      <Path.Fill>
         <SolidColorBrush Color="#0000FF" />
      </Path.Fill>
      <Path.Data>
         <PathGeometry>
            <PathFigure StartPoint="200,60" IsClosed="true">
               <PolyLineSegment Points="400,160 400,60 200,160" />
            </PathFigure>
         </PathGeometry>
      </Path.Data>
   </Path>
```

This markup is rendered as follows:



end example]

18.6 Stroke Rendering

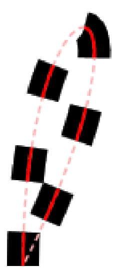
Strokes follow the contours of each segment in a path figure, as specified by the various stroke-related properties.

Contours and dashes SHOULD be rendered so that they have the same appearance as if rendered by sweeping the complete length of the contour or dash with a line segment that is perpendicular to the contour and extends with half its length to each side of the contour. All points covered by the sweep of this perpendicular line are part of the dash or contour [S11.21].

By using this sweeping definition, extreme curvatures may result in line and dash ends that are not flat when specified as flat. If any caps other than flat are specified, the caps are added to the start and end of the stroked contour or dash in the orientation of the first and last position of the line segment used for sweeping. Any render transform is applied after this step.

[*Note*: Using this definition, any geometry that is less than the value of the stroke thickness across will produce a filled area between these lines if no dashes are employed, or overlapping dashes when they are. *end note*]

Figure 18–2. Extreme curvatures and dash rendering



18.6.1 Stroke Edge Parallelization

Consumers SHOULD ensure that parallel edges of strokes appear parallel [S11.22]. Consumers can choose a suitable method to achieve this goal. [*Example*: Such methods might include anti-aliasing, sub-pixel masking, or appropriate rounding of device coordinates. *end example*]

18.6.2 Phase Control

Consumers SHOULD produce a visually consistent appearance of stroke thickness for thin lines, regardless of their orientation or how they fit on the device pixel grid [S11.23].

18.6.3 Symmetry of Stroke Drawing Algorithms

Consumers SHOULD select line and curve drawing algorithms that behave symmetrically and result in the same set of device pixels being drawn regardless of the direction of the line or curve (start point and end point exchanged) [S11.24]. In other words, a line from 0,0 to 102,50 should result in the same pixel set as a line from 102,50 to 0,0.

18.6.4 Rules for Dash Cap Rendering

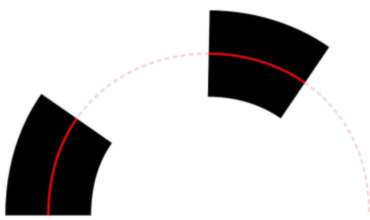
The appearance of dash caps is controlled by the **StrokeDashCap** attribute. Valid values are Flat, Square, Round, and Triangle.

18.6.4.1 Flat Dash Caps

The effective render transform of the path being stroked is used to transform the control points of the contour of the dash.

The length of the dash is the approximate distance on the curve between the two intersections of the flat lines ending the dash and the contour of the shape. The distance from the end of one dash to the start of the next dash is the specified dash gap length. Dashes with a length greater than 0 are drawn, and degenerate dashes with a length of 0 are not drawn.

Figure 18–3. Flat dash caps



18.6.4.2 Square Dash Caps

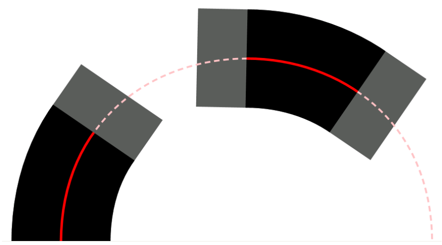
The effective render transform of the path being stroked is used to transform the control points of the contour of the dash.

The length of the dash is the approximate distance on the curve between the two *contour intersection points*, that is, the intersection of the flat line ending the dash (without the square caps attached) and the contour of the shape.

The caps are drawn as half-squares attached to the ends of the dash. The boundaries of the square caps are not curved to follow the contour, but are transformed using the effective render transform.

The distance between the contour intersection points of consecutive dashes is the specified dash gap length. Degenerate dashes with a length of 0 are drawn as squares.

Figure 18–4. Square dash caps



18.6.4.3 Round Dash Caps

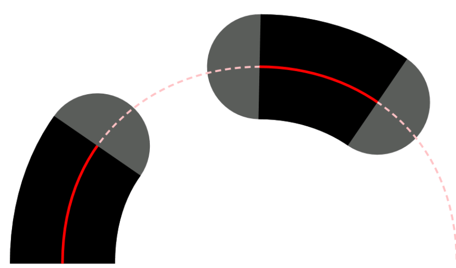
The effective render transform of the path being stroked is used to transform the control points of the contour of the dash.

The length of the dash is the approximate distance on the curve between the two contour intersection points, that is, the intersection of the flat line ending the dash (without the round caps attached) and the contour of the shape.

The caps are drawn as half-circles attached to the ends of the dash. The boundaries of the round caps are not distorted to follow the contour, but are transformed using the effective render transform.

The distance between the contour intersection points of consecutive dashes is the specified dash gap length. Degenerate dashes with a length of 0 are drawn as circles.

Figure 18–5. Round dash caps



18.6.4.4 Triangular Dash Caps

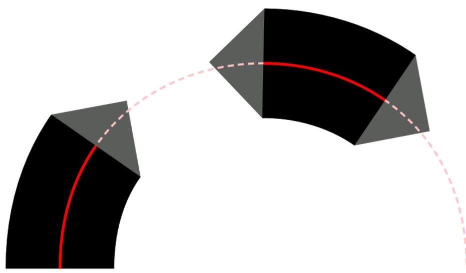
The effective render transform of the path being stroked is used to transform the control points of the contour of the dash.

The length of the dash is the approximate distance on the curve between the two contour intersection points, that is, the intersection of the flat line ending the dash (without the triangular caps attached) and the contour of the shape.

The caps are drawn as triangles attached with their base to the ends of the dash. The boundaries of the triangular caps are not distorted to follow the contour, but are transformed using the effective render transform. The height of the triangles is half of the stroke width.

The distance between the contour intersection points of consecutive dashes is the specified dash gap length. Degenerate dashes with a length of 0 are drawn as diamonds.

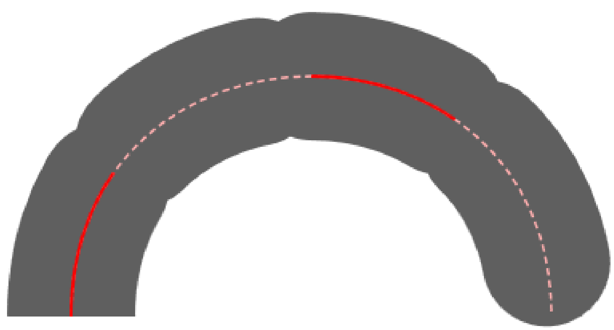
Figure 18–6. Triangular dash caps



18.6.4.5 Overlapping Dashes

It is possible to specify dash sequences with overlapping dash caps. In this circumstance, the union of the dash segments (inclusive of dash caps), is used as a mask through which the brush is applied as illustrated in Figure 18–7 with a stroke dash cap value of Round.

Figure 18–7. Overlapping dash segments



18.6.5 Rules for Line Cap Rendering

The appearance of line caps is controlled by the **StrokeStartLineCap** and **StrokeEndLineCap** attribute. Valid values are Flat, Square, Triangle, and Round. Every start line cap can be used in combination with any end line cap. Line caps are not drawn for closed paths.

The rules for line caps on curved lines are analogous to the rules for dash cap rendering. For more information, see §18.6, "Stroke Rendering," on page 288 and §18.6.4, "Rules for Dash Cap Rendering," on page 289.

Figure 18–8. Flat start line cap, flat end line cap

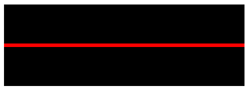


Figure 18–9. Square start line cap, square end line cap

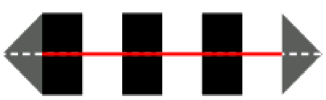
Figure 18–10. Triangular start line cap, triangular end line cap

Figure 18–11. Round start line cap, round end line cap

18.6.6 Line Caps for Dashed Strokes

If the start point of a stroke is within a dash or touches the start or end of a dash, a start line cap is appended to the stroke. Similarly, if the end point of a stroke is within a dash or touches the start or end of a dash, an end line cap is appended to the stroke.

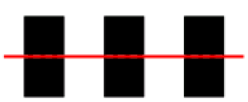
Figure 18–12. Stroke start or end point within a dash



[*Note*: Because the right-most line cap begins at the point exactly coincident with the start of the next dash in the sequence, it is rendered. *end note*]

However, if the start point of a stroke is within a gap (as can result from a **StrokeDashOffset** attribute), no start line cap is appended to the stroke. If the end point of a stroke is within a gap, no end line cap is appended to the stroke.

Figure 18–13. Stroke start or end point within a gap



18.6.7 Rules for Line Join Rendering

The appearance of line joins is controlled by the **StrokeLineJoin** attribute. Valid values are Round, Bevel, and Miter.

18.6.7.1 Round Line Joins

A **StrokeLineJoin** attribute value of Round indicates that the outer corner of the joined lines should be filled by enclosing the rounded region with its center point at the point of intersection between the two lines and a radius of one-half the stroke thickness value.

Figure 18–14. Round line join with right angle

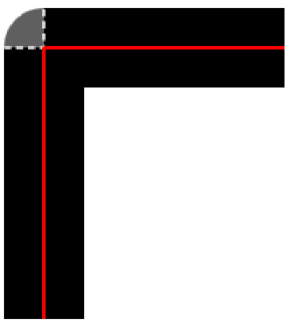


Figure 18–15. Round line join with acute angle

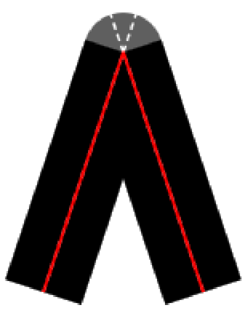
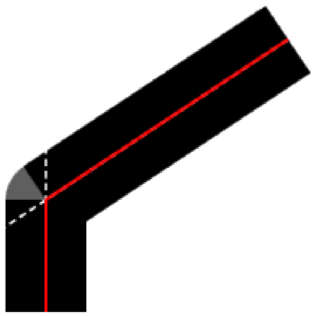


Figure 18–16. Round line join with obtuse angle



18.6.7.2 Beveled Line Joins

A **StrokeLineJoin** attribute value of Bevel indicates that the outer corner of the joined lines should be filled by enclosing the triangular region of the corner with a straight line between the outer corners of each stroke.

Figure 18–17. Beveled line join with right angle

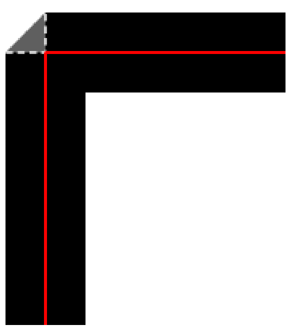


Figure 18–18. Beveled line join with acute angle

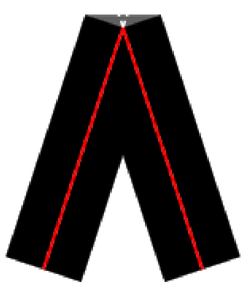
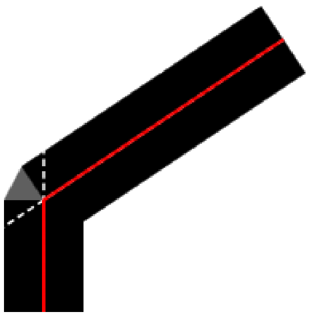


Figure 18–19. Beveled line join with obtuse angle



18.6.7.3 Mitered Line Joins

If the **StrokeLineJoin** attribute value is Miter, the value of the **StrokeMiterLimit** attribute value is also used for rendering these joins. A **StrokeLineJoin** value of Miter indicates that the region to be filled includes the intersection of the strokes projected to infinity, and then clipped at a specific distance. The intersection of the strokes is clipped at a line perpendicular to the bisector of the angle between the strokes, at the distance equal to the stroke miter limit value multiplied by half the stroke thickness value.

When drawing mitered line joins, the presence of one or more degenerate line segments between the non-degenerate line segments to be joined results in a mitered line join of only the two non-degenerate line segments with an implied **StrokeMiterLimit** attribute value of 1.0.

Figure 18–20. Mitered line join with right angle and miter limit of 1.0

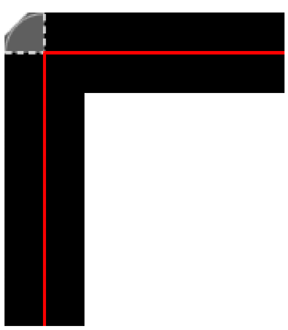


Figure 18–21. Mitered line join with acute angle and miter limit of 1.0

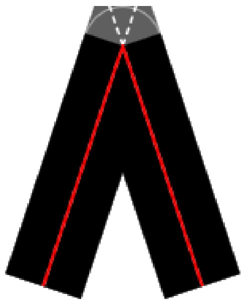


Figure 18–22. Mitered line join with obtuse angle and miter limit of 1.0

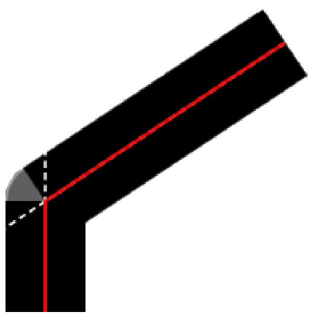


Figure 18–23. Mitered line join with right angle and miter limit of 2.0

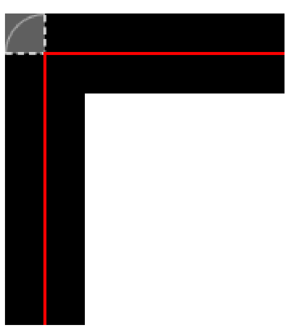


Figure 18–24. Mitered line join with acute angle and miter limit of 2.0

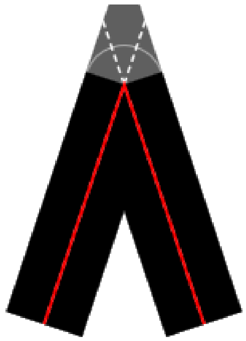
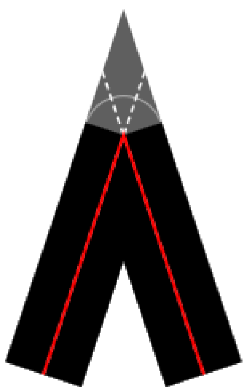


Figure 18–25. Mitered line join with acute angle and miter limit of 10.0



18.6.8 Rules for Degenerate Line and Curve Segments

Degenerate line segments (that is, where the start point and end point coincide) are not drawn.

Degenerate curve segments (where the start point, end point, and all control points coincide) are not drawn, regardless of the stroke start line cap or stroke end line cap specified.

If an open degenerate path (formed from degenerate line or curve segments) with non-flat start cap and/or non-flat end line cap is stroked, only the start line cap and/or end line cap is drawn, in the *x* direction relative to the current effective render transform (that is, as if a segment were drawn from *x*,*y* to *x*+*d*,*y*, with *d* ➔ 0).

If a closed degenerate path (formed from degenerate line or curve segments) is stroked, a circular dot with a diameter of the stroke thickness is drawn instead.

If the current render transform is an invertible matrix, consumers SHOULD perform computations on poly line segments and poly Bézier segments with sufficient accuracy to avoid producing zero-length segments [S11.25].

18.6.9 Stroking and Fill Rule

Stroking a path is independent of the fill rule. The fill rule affects the filled area only.

18.6.10 Mixing Stroked and Non-Stroked Segments

When a path figure contains multiple segments and one or more of the segments has an **IsStroked** value of false, the phase for dashes starts anew with the next stroked segment, including application of the dash offset.

When a segment is stroked and the subsequent segment has an **IsStroked** value of false, thus causing a dash to be truncated, the dash cap is drawn for both ends of the truncated dash, exactly as it would for a non-truncated dash.

18.6.11 Stroke Behavior with Multiple Path Figures

When a geometry containing multiple path figures is stroked, the phase for dashes (including application of the dash offsets) starts anew with each new path figure.

In general, for any path geometry, each path figure is drawn independently of every other path figure, so the dash array is reset for each. Dashes are also reset after every unstroked segment.

18.6.12 Consistent Nominal Stroke Width

For certain scenarios, it is desirable for producers to generate documents targeted at specific aliasing consumers with particular lines in the document indicated as hairlines or consistent-width strokes. The following recommendation allows these producers and consumers to handle these strokes consistently.

Producers MAY generate a <Path> element intended to be treated as having a consistent nominal stroke width by specifying the **StrokeDashArray** attribute and by specifying a **StrokeDashOffset** attribute value less than -1.0 times the sum of all the numbers in the **StrokeDashArray** attribute value [O11.25].

For a solid line, the producer would set the **StrokeDashArray** to the value "1 0" and the **StrokeDashOffset** to a value such as "-2". The "-2" value fulfills the restriction on the **StrokeDashOffset** value in a numerically stable manner, and the phase of the dash pattern is identical to a **StrokeDashOffset** value of "0". Values less than "-2" can be used to specify a shifted phase of the dash pattern.

A stroke using the consistent nominal stroke width convention SHOULD be rendered with a width consistent with other strokes using the convention that have the same **StrokeThickness** attribute value, and consumers aware of this convention SHOULD render such a stroke no thinner than the thinnest visible line that consumer supports without dropouts [S11.31]. See section 11.1.4, "Pixel Center Location, Pixel Placement, and Pixel Inclusion", for further considerations for rendering thin lines.

18.7 Brushes and Images

Images require the following special considerations for scaling and tile placement.

18.7.1 Small Tiles

Tiles for visual brushes and image brushes may be specified with a viewport width or height of a few device pixels, or even less than a single device pixel in size.

If both width and height are nearly zero, implementations SHOULD average the color values of the brush contents, resulting in a constant-color brush [S11.26]. [*Example*:

- A visual brush or image brush that contains a blue and white checkerboard pattern results in a solid light-blue fill as either the width or the height value approaches 0.0.
- A visual brush or image brush whose viewbox is constant-colored produces a constant-colored brush regardless of the width and height values of the viewport.

end example]

If only one of the width and height values is nearly zero, the brush should be constant-colored along lines parallel to the narrow side of the viewport. For cases such as these, implementations MAY differ [O11.21]. Producers SHOULD avoid producing such extreme cases and SHOULD NOT rely on any specific behavior when they do [S11.27].

18.7.2 Image Scaling

Source sampling SHOULD be done from the center of the pixel and should be mapped to the center of the pixel in the device-space [S11.28]. With one extent of the viewbox zero, sampling SHOULD be done along a line parallel to the non-zero side [S11.28]. With both extents of the viewbox zero, a point sample SHOULD be taken [S11.28].

When up-sampling an image presented at a lower resolution than the device resolution, bilinear filtering SHOULD be used [S11.29]. The precise source coordinates as specified by the viewbox MUST be used to place the up-sampled image tile, which is equivalent to using fractional pixels of the original source image [M11.8].

When down-sampling, at least a bilinear filter SHOULD be used [S11.30]. Consumers MAY choose to implement a more sophisticated algorithm, such as a Fant scaler, to prevent aliasing artifacts [O11.22].

18.7.3 Tile Placement

Consumers MUST precisely position the tiles specified by the image brush and visual brush. If the specified values result in fractional device pixels, the consumer MUST calculate a running placement-error delta and adjust the placement of the next tile where the delta reaches a full device pixel in order to keep the tiles from being increasingly out of phase as the expanse of the path is filled [M11.9]. Consumers MAY choose any technique desired to achieve this requirement, such as linear filtering for seams, stretching of the tile (up-sampling or down-sampling), or pre-computing multiple tiles and adjusting behavior according to how the tiles fit on a grid [O11.23].

18.7.4 Tiling Transparent Visual Brushes and Image Brushes

The contents of a visual brush's Visual property are first rendered to a temporary work canvas with an opacity of 0.0. The viewbox of the visual brush defines the tile or portion of the temporary canvas that is copied onto the specified geometry, stroke, or text. Likewise, an image specified by an image brush is also copied to a temporary work canvas. The viewbox also defines the tile for an image brush. In either case, the work canvas is scaled to properly match the edges of the tile to the size specified by the viewport.

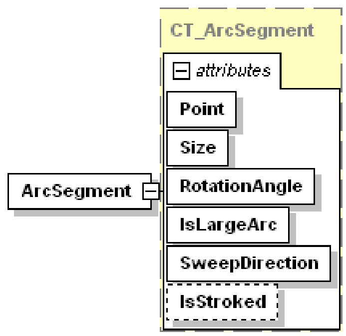
Each pixel of the resultant tile is separately blended with the background of its destination, using the alpha of each pixel. This process is repeated for each tile replication, while respecting the **TileMode** attribute value, although the temporary work canvas MAY be re-used [O11.24].

19. Elements

ArcSegment

element **ArcSegment**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
Point	ST_Point	required			Specifies the endpoint of the elliptical arc.
Size	ST_PointGE0	required			Specifies the x and y radius of the elliptical arc as an x,y pair.
RotationAngle	ST_Double	required			Indicates how the ellipse is rotated relative to the current coordinate system.
IsLargeArc	ST_Boolean	required			Determines whether the arc is drawn with a sweep of 180 or greater. Can be true or false.
SweepDirection	ST_SweepDirection	required			Specifies the direction in which the arc is drawn. Valid values are Clockwise and Counterclockwise.
IsStroked	ST_Boolean		true		Specifies whether the stroke for this segment of the path is drawn. Can be true or false.

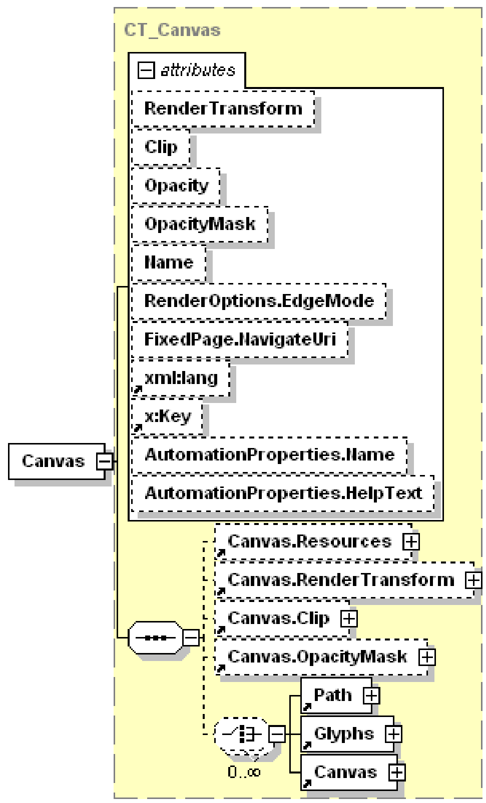
annotation

Represents an elliptical arc between two points.

Canvas

element **Canvas**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
RenderTransform	ST_RscRefMatrix				Establishes a new coordinate frame for the child and descendant elements of the canvas, such as another canvas. Also affects clip and opacity mask.
Clip	ST_RscRefAbbrGeomF				Limits the rendered region of the element.

	Opacity	ST_ZeroOne		1.0		Defines the uniform transparency of the canvas. 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 canvas 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.
	RenderOptions.EdgeMode	ST_EdgeMode				Controls how edges of paths within the canvas are rendered. The only valid value is Aliased. Omitting this attribute causes the edges to be rendered in the consumer's default manner.
	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

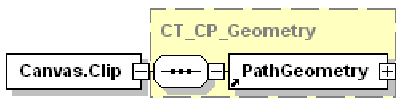
						internal to or external to the package.
	xml:lang					Specifies the default language used for the current element and for any child or descendant elements. The language is specified according to RFC 3066.
	x:Key					Specifies a name for a resource in a resource dictionary. x:Key MUST be present when the current element is defined in a resource dictionary. x:Key MUST NOT be specified outside of a resource dictionary [M3.20].
	AutomationProperties.Name	xs:string				A brief description of the <Canvas> contents for accessibility purposes, particularly if filled with a set of vector graphics and text elements intended to comprise a single vector graphic.
	AutomationProperties.HelpText	xs:string				A detailed description of the <Canvas> contents for accessibility purposes, particularly if filled with a set of graphics and text elements intended

						to comprise a single vector graphic.
annotation	Groups <FixedPage> descendant elements together.					

For more information, see §10.4, "<Canvas> Element," on page 61.

Canvas.Clip

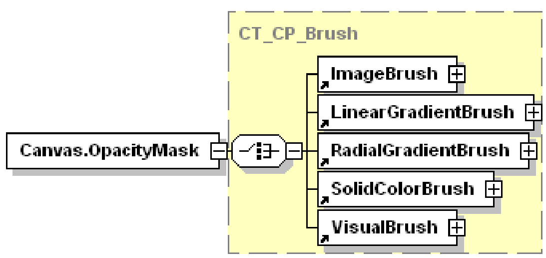
element **Canvas.Clip**

diagram	
annotation	Limits the rendered region of the element.

For more information, see §14.3, "Clipping," on page 182.

Canvas.OpacityMask

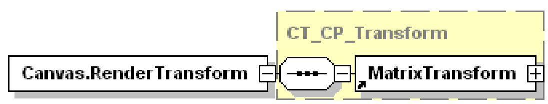
element **Canvas.OpacityMask**

diagram	
annotation	Specifies a mask of alpha values that is applied to the canvas in the same fashion as the Opacity attribute, but allowing different alpha values for different areas of the element.

For more information, see §14.5.1, "<Canvas.OpacityMask> Element," on page 200.

Canvas.RenderTransform

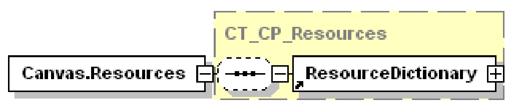
element **Canvas.RenderTransform**

diagram	
annotation	Establishes a new coordinate frame for the child and descendant elements of the canvas, such as another canvas. Also affects clip and opacity mask.

For more information, see §14.4, "Positioning Content," on page 185.

Canvas.Resources

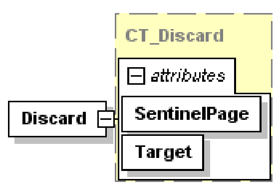
element **Canvas.Resources**

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

For more information, see §14.2, "Resources and Resource References," on page 172.

Discard

element **Discard**

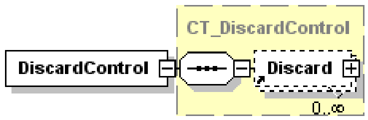
diagram	
attributes	see below
annotation	Identifies a resource that can be safely discarded by a resource-constrained consumer.

Name	Type	Use	Default	Fixed	Annotation
SentinelPage	xs:anyURI	required			The first fixed page that no longer needs the identified resource in order to be processed.
Target	xs:anyURI	required			The resource that can be safely discarded.

For more information, see §17.1.4.1.2, "<Discard> Element."

1 **DiscardControl**

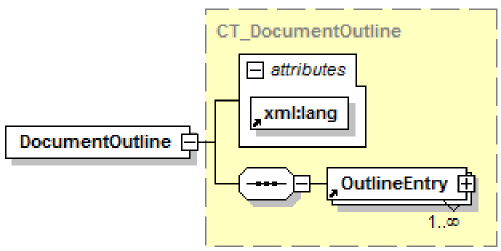
2 element **DiscardControl**

diagram	
annotation	Contains a list of resources that are safe for a consumer to discard.

3 For more information, see §17.1.4.1.1, “<DiscardControl> Element.”

4 **DocumentOutline**

5 element **DocumentOutline**

diagram	
attributes	see below
annotation	Specifies a list of meaningful indices into the XPS Document, similar to a table of contents, or to external URIs, such as web addresses.

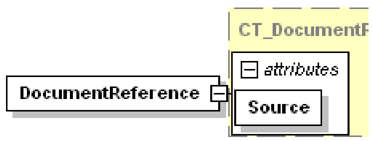
attributes:

Name	Type	Use	Default	Fixed	Annotation
xml:lang		required			This attribute specifies the default language used for any child element contained within the current element or nested child elements. The language is specified according to IETF RFC 3066.

6 For more information, see §16.1.1.3, “<DocumentOutline> Element,” on page 226.

DocumentReference

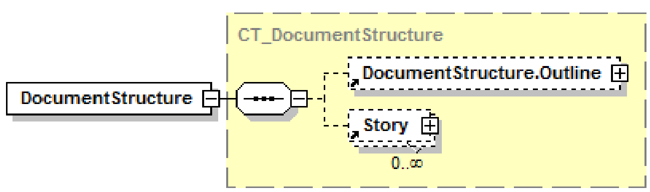
element **DocumentReference**

<table>
<tr><td>diagram</td><td>
</td></tr>
<tr><td>attributes</td><td><table><tr><td>Name</td><td>Type</td><td>Use</td><td>Default</td><td>Fixed</td><td>Annotation</td></tr><tr><td>Source</td><td>xs:anyURI</td><td>required</td><td></td><td></td><td>Specifies the URI of the fixed document content. The specified URI MUST refer to a FixedDocument part within the XPS Document [M3.2].</td></tr></table></td></tr>
<tr><td>annotation</td><td>Contains a reference to a FixedDocument part.</td></tr>
</table>

For more information, see §10.1.1, “<DocumentReference,” on page 49.

DocumentStructure

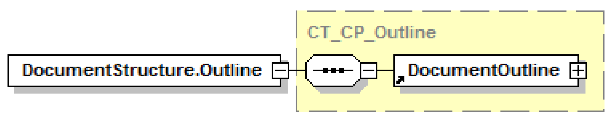
element **DocumentStructure**

<table>
<tr><td>diagram</td><td>
</td></tr>
<tr><td>annotation</td><td>The root element of the DocumentStructure part.</td></tr>
</table>

For more information, see §16.1.1.1, “<DocumentStructure> Element,” on page 224.

DocumentStructure.Outline

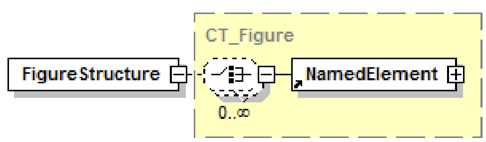
element **DocumentStructure.Outline**

diagram	
annotation	Contains a structured document outline that provides a list of links into the document contents or external sites.

For more information see §16.1.1.2, "<DocumentStructure.Outline> Element," on page 226.

FigureStructure

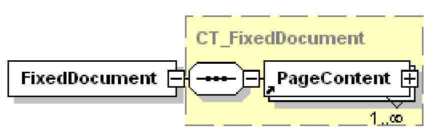
element **FigureStructure**

diagram	
annotation	Groups the named elements that constitute a single drawing or diagram.

For more information, see §16.1.2.12, "<FigureStructure> Element," on page 242.

FixedDocument

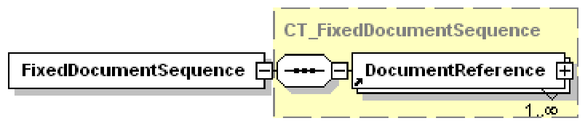
element **FixedDocument**

diagram	
annotation	Binds an ordered sequence of fixed pages together into a single multi-page document.

For more information, see §10.2, "<FixedDocument> Element," on page 50.

FixedDocumentSequence

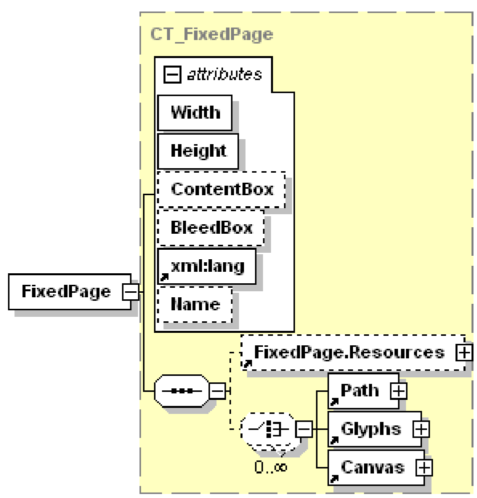
element **FixedDocumentSequence**

diagram	
annotation	Specifies a sequence of fixed documents.

For more information, see §10.1, “<FixedDocumentSequence> Element,” on page 49.

FixedPage

element **FixedPage**

diagram	
attributes	(see below)

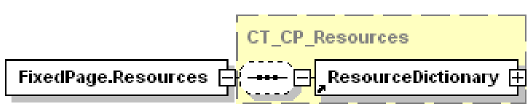
Name	Type	Use	Default	Fixed	Annotation
Width	ST_GEOne	required			Width of the page, expressed as a real number in units of the effective coordinate space.
Height	ST_GEOne	required			Height of the page, expressed as a real number in units of the effective coordinate space.
ContentBox	ST_ContentBox				Specifies the area of the page

						containing imageable content that is to be fit within the imageable area when printing or viewing. Contains a list of four coordinate values (ContentOriginX, ContentOriginY, ContentWidth, ContentHeight), expressed as comma-separated real numbers. Specifying a value is RECOMMENDED [S3.1]. If omitted, the default value is (0,0,Width,Height).
	BleedBox	ST_BleedBox				Specifies the area including crop marks that extends outside of the physical page. Contains a list of four coordinate values (BleedOriginX, BleedOriginY, BleedWidth, BleedHeight), expressed as comma-separated real numbers. If omitted, the default value is (0,0,Width,Height).
	xml:lang		required			Specifies the default language used for the current element and for any child or descendant elements. The language is specified according to RFC 3066.
	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.
annotation	Contains markup that describes the rendering of a single page of content.					

For more information, see §10.3, "<FixedPage> Element," on page 53.

FixedPage.Resources

element **FixedPage.Resources**

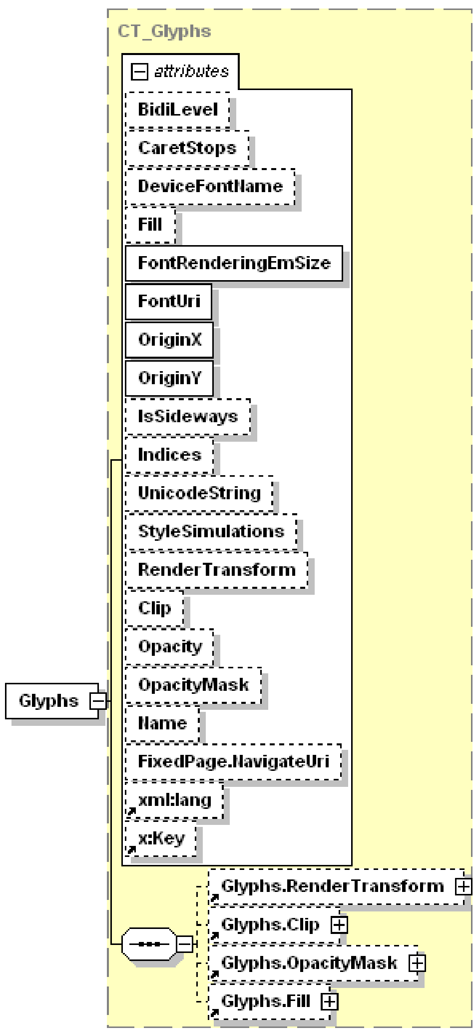
diagram	
annotation	Contains the resource dictionary for the <FixedPage> element.

For more information, see §14.2, "Resources and Resource References," on page 172.

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 may 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

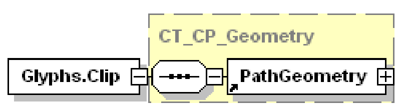
						internal to or external to the package.
	xml:lang					Specifies the default language used for the current element. The language is specified according to RFC 3066.
	x:Key					Specifies a name for a resource in a resource dictionary. x:Key MUST be present when the current element is defined in a resource dictionary. x:Key MUST NOT be specified outside of a resource dictionary [M5.3].
annotation	Represents a run of text from a single font.					

For more information, see:

- Section 12.1, "<Glyphs> Element," on page 98
- Section 9.1.7, "Font Parts," on page 31
- Section 12.1.3, "Indices Attribute," on page 109

Glyphs.Clip

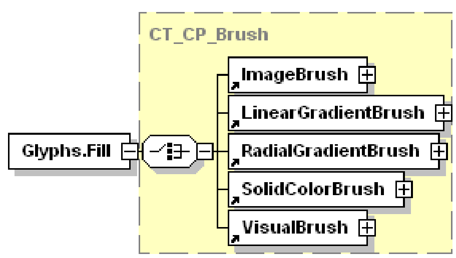
element **Glyphs.Clip**

diagram	
annotation	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.

For more information, see §14.3, "Clipping," on page 182.

Glyphs.Fill

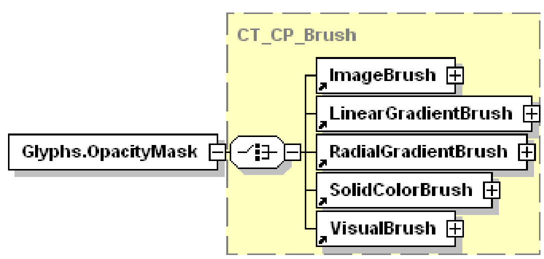
element **Glyphs.Fill**

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

For more information, see §12.2, ”<Glyphs.Fill> Element.”

Glyphs.OpacityMask

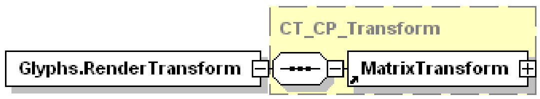
element **Glyphs.OpacityMask**

diagram	
annotation	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.

For more information, see §14.5.3, “<Glyphs.OpacityMask> Element,” on page 202.

Glyphs.RenderTransform

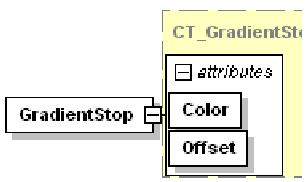
element **Glyphs.RenderTransform**

diagram	
annotation	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.

For more information, see §14.4, “Positioning Content,” on page 185.

GradientStop

element **GradientStop**

diagram	
attributes	see table below
annotation	Indicates a location and range of color progression for rendering a gradient.

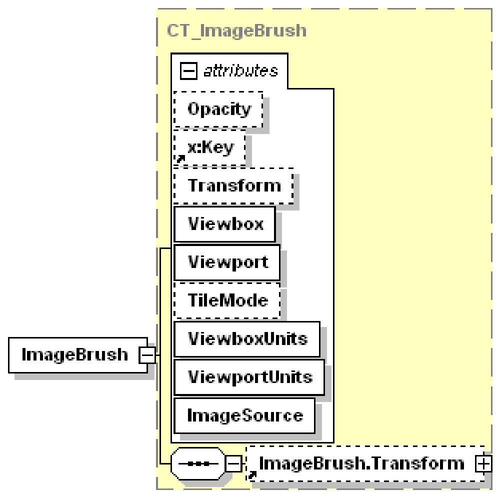
Name	Type	Use	Default	Fixed	Annotation
Color	ST_Color	required			Specifies the gradient stop color. An sRGB color value specified as a 6-digit hexadecimal number (#RRGGBB) or an extended color.
Offset	ST_Double	required			Specifies the gradient offset. The offset indicates a point along the progression of the gradient at which a color is specified. Colors between gradient offsets in the progression are interpolated.

For more information, see §13.7, “<GradientStop> Element,” on page 166.

ImageBrush

element **ImageBrush**

diagram



attributes

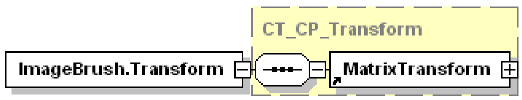
Name	Type	Use	Default	Fixed	Annotation
Opacity	ST_ZeroOne		1.0		Defines the uniform transparency of the brush fill. Values range from 0 (fully transparent) to 1 (fully opaque), inclusive. Values outside of this range are invalid.
x:Key					Specifies a name for a resource in a resource dictionary. x:Key MUST be present when the current element is defined in a resource dictionary. x:Key MUST NOT be specified outside of a resource dictionary [M6.2].
Transform	ST_RscRefMatrix				Describes the matrix transformation applied to the coordinate space of the brush. The Transform property is concatenated with the current effective render transform to yield an effective render transform local to the brush. The viewport for the brush is transformed using the local effective render transform.

	Viewbox	ST_ViewBox	required			Specifies the position and dimensions of the brush's source content. Specifies four comma-separated real numbers (x, y, Width, Height), where width and height are non-negative. The dimensions specified are relative to the image's physical dimensions expressed in units of 1/96". The corners of the viewbox are mapped to the corners of the viewport, thereby providing the default clipping and transform for the brush's source content.
	Viewport	ST_ViewBox	required			Specifies the region in the containing coordinate space of the prime brush tile that is (possibly repeatedly) applied to fill the region to which the brush is applied. Specifies four comma-separated real numbers (x, y, Width, Height), where width and height are non-negative. The alignment of the brush pattern is controlled by adjusting the x and y values.
	TileMode	ST_TileMode		None		Specifies how contents will be tiled in the filled region. Valid values are None, Tile, FlipX, FlipY, and FlipXY.
	ViewboxUnits	ST_ViewUnits	required		Absolute	Specifies the relationship of the viewbox coordinates to the containing coordinate space.
	ViewportUnits	ST_ViewUnits	required		Absolute	Specifies the relationship of the viewport coordinates to the containing coordinate space.
	ImageSource	ST_UriCtxBmp	required			Specifies the URI of an image resource or a combination of the URI of an image resource a color profile resource. See the Color clause for important details. The URI MUST refer to parts in the package [M2.1].
annotation	Fills a region with an image.					

1 For more information, see §13.2, “<ImageBrush> Element,” on page 127.

2 **ImageBrush.Transform**

3 element **ImageBrush.Transform**

diagram	
annotation	Describes the matrix transformation applied to the coordinate space of the brush. The Transform property is concatenated with the current effective render transform to yield an effective render transform local to the brush. The viewport for the brush is transformed using the local effective render transform.

4 For more information, see §14.4, “Positioning Content,” on page 185.

5 **Intent**

6 element **SignatureDefinitionType/Intent**

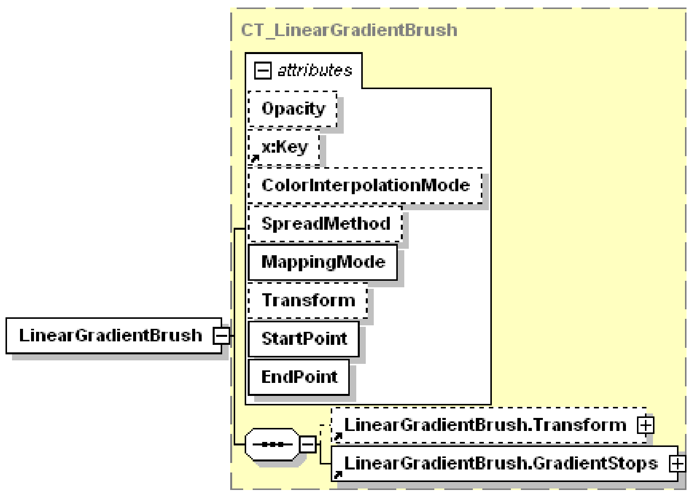
diagram	
annotation	A string that represents the intent to which the signing party agrees when signing the document.

7 For more information, see §17.2.2.4, “<Intent> Element.”

LinearGradientBrush

element **LinearGradientBrush**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
Opacity	ST_ZeroOne		1.0		Defines the uniform transparency of the linear gradient. Values range from 0 (fully transparent) to 1 (fully opaque), inclusive. Values outside of this range are invalid.
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 [M6.5].
ColorInterpolationMode	ST_ClrIntMode		SRgbLinear Interpolation		Specifies the gamma function for color interpolation. The gamma adjustment should not be applied to

						the alpha component, if specified. Valid values are SRgbLinearInterpolation and ScRgbLinearInterpolation.
	SpreadMethod	ST_SpreadMethod		Pad		Describes how the brush should fill the content area outside of the primary, initial gradient area. Valid values are Pad, Reflect and Repeat.
	MappingMode	ST_MappingMode	required		Absolute	Specifies that the start point and end point are defined in the effective coordinate space (includes the Transform attribute of the brush).
	Transform	ST_RscRefMatrix				Describes the matrix transformation applied to the coordinate space of the brush. The Transform property on a brush is concatenated with the current effective render transform to yield an effective render transform local to the brush. The start point and end point are transformed using the local effective render transform.
	StartPoint	ST_Point	required			Specifies the starting point of the linear gradient.
	EndPoint	ST_Point	required			Specifies the end point of the linear gradient. The linear gradient brush interpolates the colors from the start point to the end point, where the start point represents an offset of 0, and the EndPoint

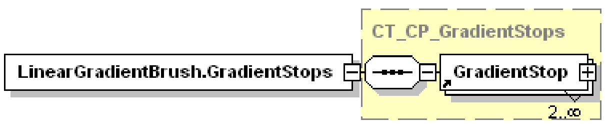
						represents an offset of 1. The Offset attribute value specified in a GradientStop element relates to the 0 and 1 offsets defined by the start point and end point.
annotation	Fills a region with a linear gradient.					

For more information, see

- Section 13.5, "<LinearGradientBrush> Element," on page 151
- 15, "Color"

LinearGradientBrush.GradientStops

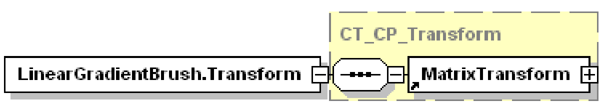
element **LinearGradientBrush.GradientStops**

diagram	
annotation	Holds a sequence of GradientStop elements.

For more information, see §13.5.2, "<LinearGradientBrush.GradientStops> Element," on page 157.

LinearGradientBrush.Transform

element **LinearGradientBrush.Transform**

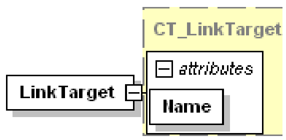
diagram	
annotation	Describes the matrix transformation applied to the coordinate space of the brush. The Transform property is concatenated with the current effective render transform to yield an effective render transform local to the brush. The start point and end point are transformed using the local effective render transform.

For more information, see §14.4.8, "<LinearGradientBrush.Transform> Element," on page 197.

LinkTarget

element **LinkTarget**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
Name	ST_Name	required			Contains a string value that identifies the current element as a named, addressable point in the document for the purpose of hyperlinking.

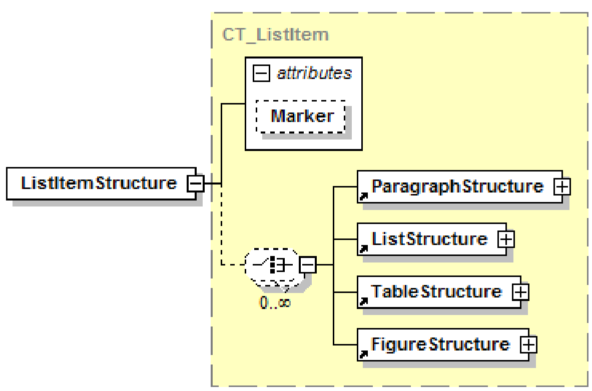
annotation: Specifies an addressable point on the page.

For more information, see §10.2.3, “<LinkTarget> Element,” on page 52.

ListItemStructure

element **ListItemStructure**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
Marker	ST_NameUnique	optional			The named element that represents the marker for this list items, such as a bullet, number, or image.

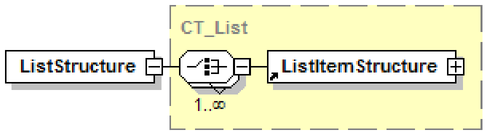
annotation: Describes a single structural block. These structural blocks are grouped together in a list.

For more information, see §16.1.2.11, “<ListItemStructure> Element,” on page 242.

ListStructure

element **ListStructure**

diagram



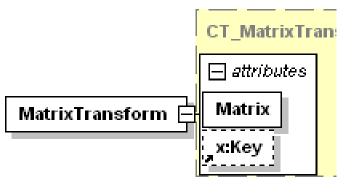
annotation	Contains a collection of items that are group together in a list.

For more information, see §16.1.2.10, “<ListStructure> Element,” on page 241.

MatrixTransform

element **MatrixTransform**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
Matrix	ST_Matrix	required			Specifies the matrix structure that defines the transformation.
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 [M7.11].

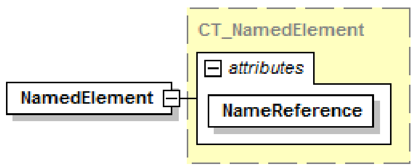
annotation	Creates an arbitrary affine matrix transformation that manipulates objects or coordinate systems in a two-dimensional plane.

For more information, see §14.4.1, “<MatrixTransform> Element,” on page 185.

1 **NamedElement**

2 element **NamedElement**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
NameReference	ST_Name	required			Identifies the named element in the FixedPage part markup that is referenced by the document structure markup.

annotation

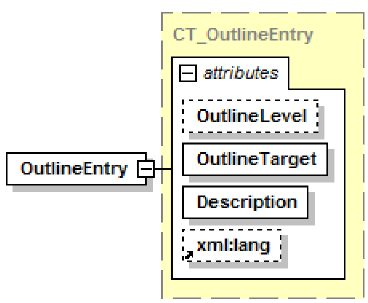
All document structure is related to the fixed page markup using this element. The <NamedElement> points to a single markup element contained in the fixed page markup.

3 For more information, see §16.1.2.13, "<NamedElement> Element," on page 243.

4 **OutlineEntry**

5 element **OutlineEntry**

diagram



attributes

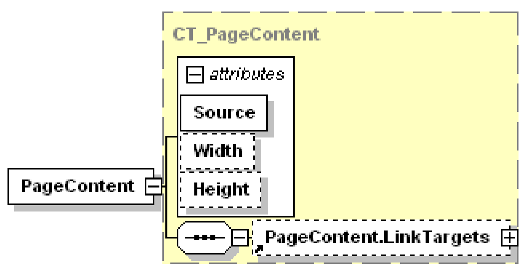
Name	Type	Use	Default	Fixed	Annotation
OutlineLevel	ST_IntGEOne	optional	1		A description of the level where the outline entry exists in the hierarchy. A value of 1 is the root.
OutlineTarget	xs:anyURI	required			The URI to which the outline entry is linked. This may be a URI to a named element within the document or an

						external URI, such as a website. It can be used as a hyperlink destination.
	Description	xs:string	required			The friendly text associated with this outline entry.
	xml:lang		optional			This attribute specifies the default language used for any child element contained within the current element or nested child elements. The language is specified according to IETF RFC 3066.
annotation	Represents an index to a specific location in the document.					

For more information, see §16.1.1.4, "<OutlineEntry> Element," on page 227.

PageContent

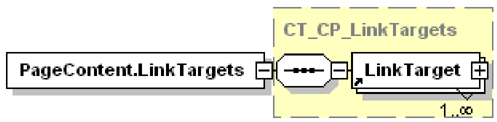
element **PageContent**

diagram						
attributes	Name	Type	Use	Default	Fixed	Annotation
	Source	xs:anyURI	required			Specifies a URI that refers to the page content, held in a distinct part within the package. The content identified MUST be a FixedPage part within the XPS Document [M3.5].
	Width	ST_GEOne				Typical width of pages contained in the page content.
	Height	ST_GEOne				Typical height of pages contained in the page content.
annotation	Defines a reference from a fixed document to a part that contains a <FixedPage> element.					

For more information, see §10.2.1, "<PageContent> Element," on page 50.

PageContent.LinkTargets

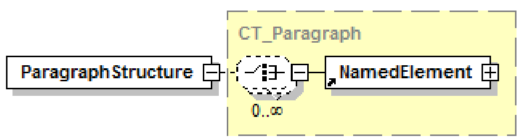
element **PageContent.LinkTargets**

diagram	
annotation	Contains a collection of <LinkTarget> elements, each of which is addressable via hyperlink.

For more information, see §10.2.2, “<PageContent.LinkTargets> Element,” on page 51.

ParagraphStructure

element **ParagraphStructure**

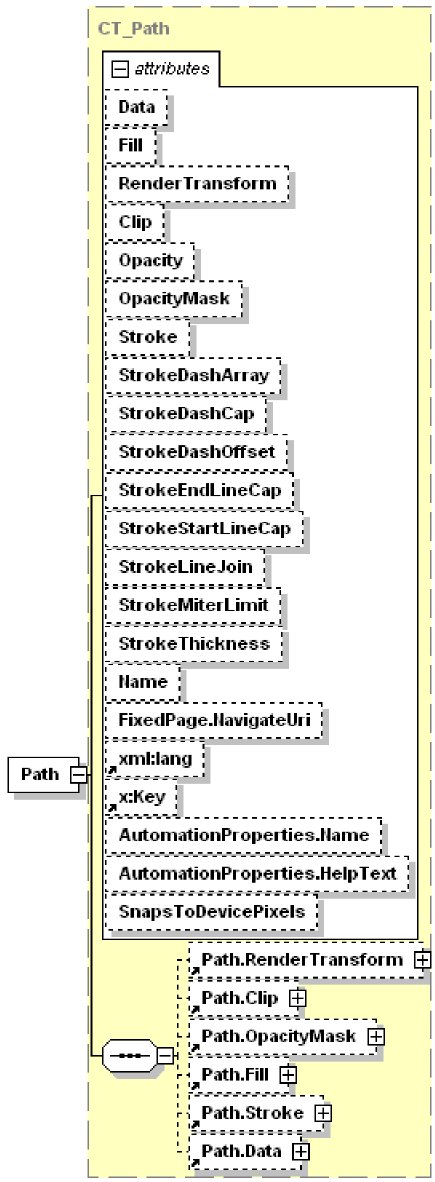
diagram	
annotation	Contains the named elements that constitute a single paragraph.

For more information, see §16.1.2.5, “<ParagraphStructure> Element,” on page 239.

Path

element **Path**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation

	Data	ST_RscRefAbbrGeomF				Describes the geometry of the path.
	Fill	ST_RscRefColor				Describes the brush used to paint the geometry specified by the Data property of the path.
	RenderTransform	ST_RscRefMatrix				Establishes a new coordinate frame for all attributes of the path and for all child elements of the path, such as the geometry defined by the <Path.Data> property element.
	Clip	ST_RscRefAbbrGeomF				Limits the rendered region of the element.
	Opacity	ST_ZeroOne		1.0		Defines the uniform transparency of the path 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 path in the same fashion as the Opacity attribute, but allowing different alpha values for different areas of the element.

	Stroke	ST_RscRefColor				Specifes the brush used to draw the stroke.
	StrokeDashArray	ST_EvenArrayPos				Specifies the length of dashes and gaps of the outline stroke. These values are specified as multiples of the stroke thickness as a space-separated list with an even number of non-negative values. When a stroke is drawn, the dashes and gaps specified by these values are repeated to cover the length of the stroke. If this attribute is omitted, the stroke is drawn solid, without any gaps.
	StrokeDashCap	ST_DashCap		Flat		Specifies how the ends of each dash are drawn. Valid values are Flat, Round, Square, and Triangle.
	StrokeDashOffset	ST_Double		0.0		Adjusts the start point for repeating the dash array pattern. If this value is omitted, the dash array aligns with the origin of the stroke. Values are specified as multiples of the stroke thickness.
	StrokeEndLineCap	ST_LineCap		Flat		Defines the shape of the end of the

						last dash in a stroke. Valid values are Flat, Square, Round, and Triangle.
	StrokeStartLineCap	ST_LineCap		Flat		Defines the shape of the beginning of the first dash in a stroke. Valid values are Flat, Square, Round, and Triangle.
	StrokeLineJoin	ST_LineJoin		Miter		Specifies how a stroke is drawn at a corner of a path. Valid values are Miter, Bevel, and Round. If Miter is selected, the value of StrokeMiterLimit is used in drawing the stroke.
	StrokeMiterLimit	ST_GEOne		10.0		The ratio between the maximum miter length and half of the stroke thickness. This value is significant only if the StrokeLineJoin attribute specifies Miter.
	StrokeThickness	ST_GEZero		1.0		Specifies the thickness of a stroke, in units of the effective coordinate space (includes the path's render transform). The stroke is drawn on top of the boundary of the geometry specified by the <Path> element's Data property. Half of the

						StrokeThickness extends outside of the geometry specified by the Data property and the other half extends inside of the geometry.
	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. Can 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 and for any child or descendant elements. 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

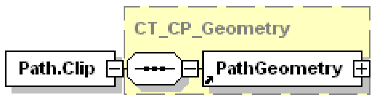
						dictionary [M4.1].
	AutomationProperties.Name	xs:string				A brief description of the <Path> for accessibility purposes, particularly if filled with an <ImageBrush>.
	AutomationProperties.HelpText	xs:string				A detailed description of the <Path> for accessibility purposes, particularly if filled with an <ImageBrush>.
	SnapsToDevicePixels	ST_Boolean				On Anti-aliasing consumers controls if control points snap to the nearest device pixels. Valid values are ‘false’ and ‘true’. Consumers MAY ignore this attribute [O4.1].
annotation	Defines a single graphical effect to be rendered to the page. It paints a geometry with a brush and draws a stroke around it.					

For more information, see

- Section 11.1, “<Path> Element,” on page 68
- Section **11.2.3**, “**Abbreviated Geometry Syntax**,” on page 90

Path.Clip

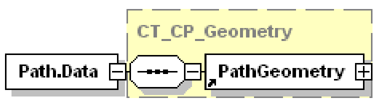
element **Path.Clip**

diagram	
annotation	Limits the rendered region of the element.

For more information, see §14.3, “Clipping,” on page 182.

Path.Data

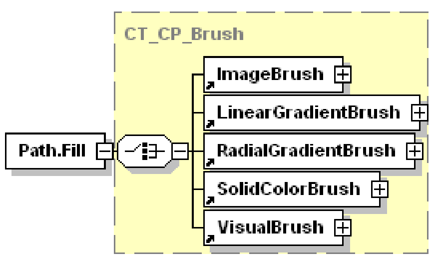
element **Path.Data**

diagram	
annotation	Describes the geometry of the path.

For more information, see §11.1.1, “<Path.Data> Element,” on page 74.

Path.Fill

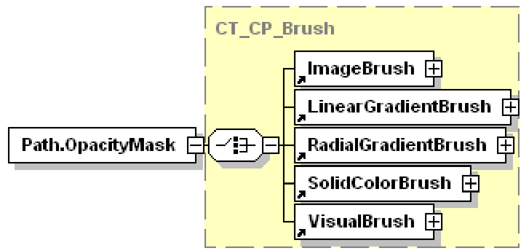
element **Path.Fill**

diagram	
annotation	Describes the brush used to paint the geometry specified by the Data property of the path.

For more information, see §11.1.2, “<Path.Fill> Element,” on page 75.

Path.OpacityMask

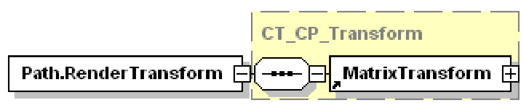
element **Path.OpacityMask**

diagram	
annotation	Specifies the mask of alpha values that is applied to the path in the same fashion as the Opacity attribute, but allowing different alpha values for different areas of the element.

For more information, see §14.5.2, “<Path.OpacityMask,” on page 201.

Path.RenderTransform

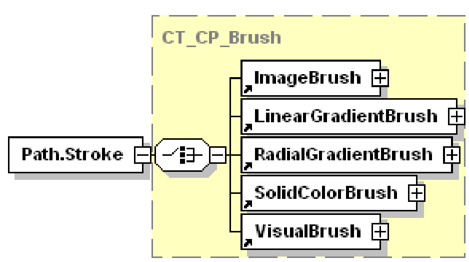
element **Path.RenderTransform**

diagram	
annotation	Establishes a new coordinate frame for all attributes of the path and for all child elements of the path, such as the geometry defined by the <Path.Data> property element.

For more information, see §14.4, “Positioning Content,” on page 185.

Path.Stroke

element **Path.Stroke**

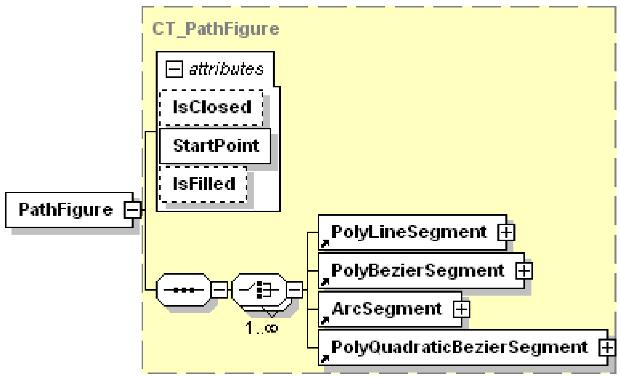
diagram	
annotation	Specifies the brush used to draw the stroke.

For more information, see §11.1.3, “<Path.Stroke> Element,” on page 76.

PathFigure

element **PathFigure**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
IsClosed	ST_Boolean		false		Specifies whether the path is closed. If set to true, the stroke is drawn "closed," that is, the last point in the last segment of the path figure is connected with the point specified in the StartPoint attribute, otherwise the stroke is drawn "open," and the last point is not connected to the start point. Only applicable if the path figure is used in a <Path> element that specifies a stroke.
StartPoint	ST_Point	required			Specifies the starting point for the first segment of the path figure.
IsFilled	ST_Boolean		true		Specifies whether the path figure is used in computing the area of the containing path geometry. Can be true or false. When set to false, the path figure is considered only for stroking.

annotation

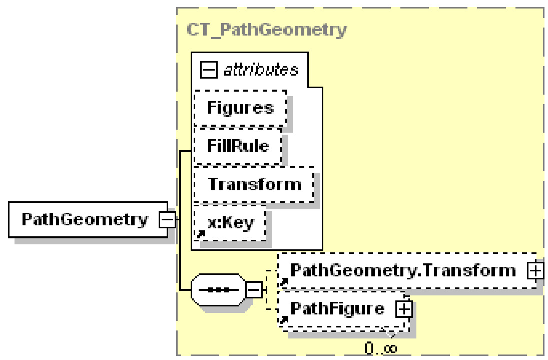
Specifies a set of one or more segment elements defining a closed region.

For more information, see §11.2.2.1, “<PathFigure> Element.”

PathGeometry

element **PathGeometry**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
Figures	ST_AbbrGeom				Describes the geometry of the path.
FillRule	ST_FillRule		EvenOdd		Specifies how the intersecting areas of geometric shapes are combined to form a region. Valid values are EvenOdd and NonZero.
Transform	ST_RscRefMatrix				Specifies the local matrix transformation that is applied to all child and descendant elements of the path geometry before it is used for filling, clipping, or stroking.
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 [M4.2].

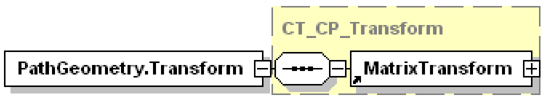
annotation

Contains a set of <PathFigure> elements.

For more information, see §11.2.1.1, "<PathGeometry> Element."

PathGeometry.Transform

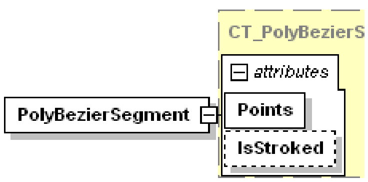
element **PathGeometry.Transform**

diagram	
annotation	Specifies the local matrix transformation that is applied to all child and descendant elements of the path geometry before it is used for filling, clipping, or stroking.

For more information, see §14.4.5, “<PathGeometry.Transform> Element,” on page 192.

PolyBezierSegment

element **PolyBezierSegment**

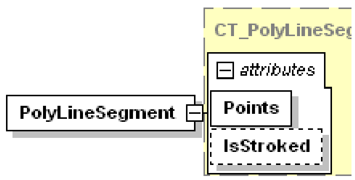
diagram	
attributes	see table below
annotation	A series of Bézier segments.

Name	Type	Use	Default	Fixed	Annotation
Points	ST_Points	required			Specifies control points for multiple Bézier segments. Coordinate values within each pair are comma-separated and additional whitespace may appear. Coordinate pairs are separated from other coordinate pairs by whitespace.
IsStroked	ST_Boolean		true		Specifies whether the stroke for this segment of the path is drawn. Can be true or false.

PolyLineSegment

element **PolyLineSegment**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
Points	ST_Points	required			Specifies a set of coordinates for the multiple segments that define the poly line segment. Coordinate values within each pair are comma-separated and additional whitespace may appear. Coordinate pairs are separated from other coordinate pairs by whitespace.
IsStroked	ST_Boolean		true		Specifies whether the stroke for this segment of the path is drawn. Can be true or false.

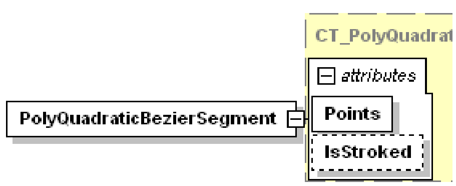
annotation

Specifies a set of points between which lines are drawn.

PolyQuadraticBezierSegment

element **PolyQuadraticBezierSegment**

diagram



attributes

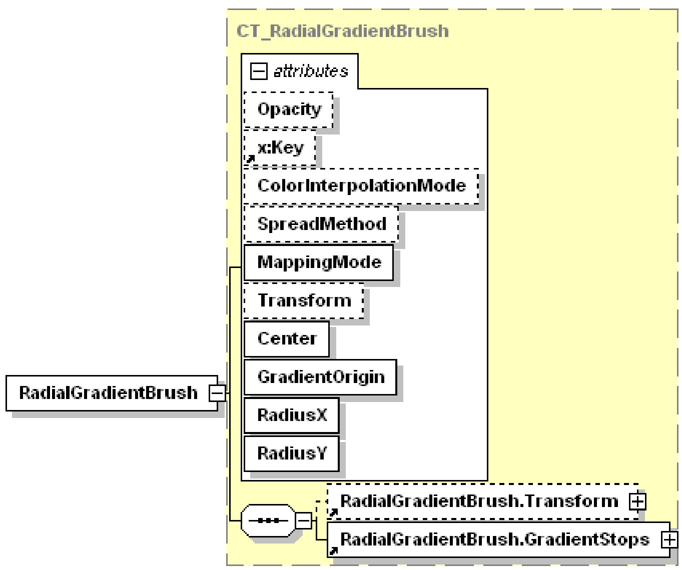
Name	Type	Use	Default	Fixed	Annotation
Points	ST_Points	required			Specifies control points for multiple quadratic Bézier segments. Coordinate values within each pair are comma-separated and additional whitespace may appear. Coordinate pairs are separated from other coordinate pairs by whitespace.
IsStroked	ST_Boolean		true		Specifies whether the stroke for this segment of the path is drawn. Can be true or false.

annotation	A series of quadratic Bézier segments.

RadialGradientBrush

element **RadialGradientBrush**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
Opacity	ST_ZeroOne		1.0		Defines the uniform transparency of the radial gradient. Values range from 0 (fully transparent) to 1 (fully opaque), inclusive. Values outside of this range are invalid.
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 [M6.6].

	ColorInterpolationMode	ST_ClrIntMode		SRgbLinear Interpolation		Specifies the gamma function for color interpolation for sRGB colors. The gamma adjustment should not be applied to the alpha component, if specified. Valid values are SRgbLinearInterpolation and ScRgbLinearInterpolation.
	SpreadMethod	ST_Spread Method		Pad		Describes how the brush should fill the content area outside of the primary, initial gradient area. Valid values are Pad, Reflect and Repeat.
	MappingMode	ST_Mapping Mode	required		Absolute	Specifies that center, x radius, and y radius are defined in the effective coordinate space (includes the Transform attribute of the brush).
	Transform	ST_RscRef Matrix				Describes the matrix transformation applied to the coordinate space of the brush. The Transform property is concatenated with the current effective render transform to yield an effective render transform local to the brush. The ellipse defined by the center, gradient origin, x radius, and y radius values is transformed using the local effective render transform.
	Center	ST_Point	required			Specifies the center point of the radial gradient (that is, the center of the ellipse). The radial gradient brush interpolates the colors from the gradient

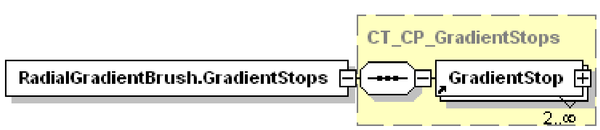
						origin to the circumference of the ellipse. The circumference is determined by the center and the radii.
	GradientOrigin	ST_Point	required			Specifies the origin point of the radial gradient.
	RadiusX	ST_GEZero	required			Specifies the radius in the x dimension of the ellipse which defines the radial gradient.
	RadiusY	ST_GEZero	required			Specifies the radius in the y dimension of the ellipse which defines the radial gradient.
annotation	Fills a region with a radial gradient.					

For more information, see:

- Section 13.6, “<RadialGradientBrush> Element,” on page 158
- 15, “Color”

RadialGradientBrush.GradientStops

element **RadialGradientBrush.GradientStops**

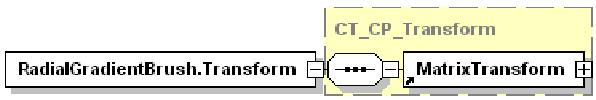
diagram	
annotation	Holds a sequence of <GradientStop> elements.

For more information, see §13.6.2, “<RadialGradientBrush.GradientStops> Element,” on page 165.

RadialGradientBrush.Transform

element **RadialGradientBrush.Transform**

diagram



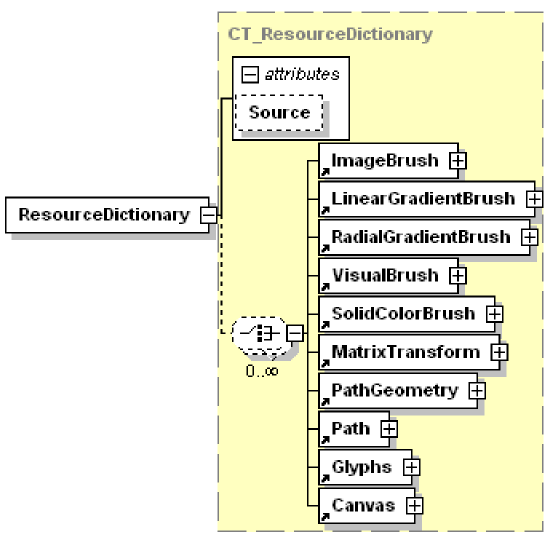
annotation	Describes the matrix transformation applied to the coordinate space of the brush. The Transform property is concatenated with the current effective render transform to yield an effective render transform local to the brush. The center, gradient origin, x radius, and y radius are transformed using the local effective render transform.

For more information, see §14.4.9, “<RadialGradientBrush.Transform> Element,” on page 198.

ResourceDictionary

element **ResourceDictionary**

diagram



attributes

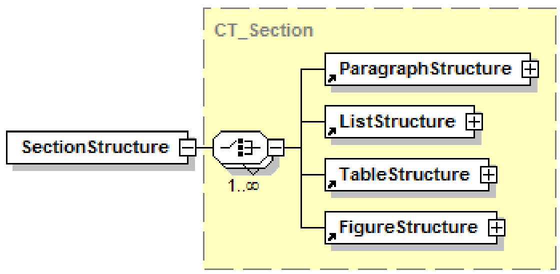
Name	Type	Use	Default	Fixed	Annotation
Source	xs:anyURI				Specifies the URI of a part containing markup for a resource dictionary. The URI MUST refer to a part in the package [M2.1].

annotation	Defines a set of reusable resource definitions that can be used as property values in the fixed page markup.

For more information, see §14.2, "Resources and Resource References," on page 172.

SectionStructure

element **SectionStructure**

diagram	
annotation	Provides an arbitrary grouping of content structural markup elements.

For more information, see §16.1.2.4, "<SectionStructure> Element," on page 239.

SignBy

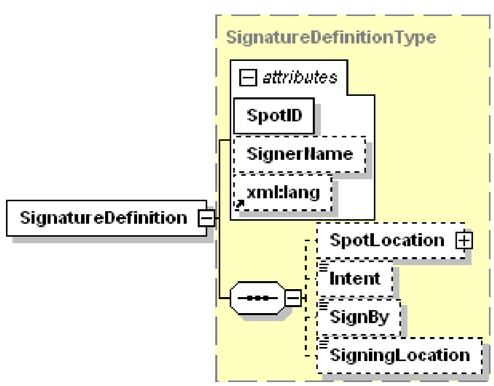
element **SignatureDefinitionType/SignBy**

diagram	
annotation	The date and time by which the requested party is to sign the XPS Document.

For more information, see §17.2.2.5, "<SignBy> Element."

SignatureDefinition

element **SignatureDefinitionsType/SignatureDefinition**

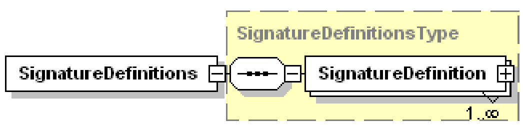
diagram	

attributes	Name	Type	Use	Default	Fixed	Annotation
	SpotID	xs:ID	required			A globally unique identifier for this signature spot.
	SignerName	xs:string				A string representing the identity of the individual who is requested to sign the XPS Document, or the name of the individual who has signed the XPS Document.
	xml:lang					Specifies the language used for the current element and its descendants. The language is specified according to RFC 3066.
annotation	A single signature definition.					

For more information, see §17.2.2.2, “<SignatureDefinition> Element.”

SignatureDefinitions

element **SignatureDefinitions**

diagram	
annotation	The root element for the SignatureDefinitions part.

For more information, see §17.2.2.1, “<SignatureDefinitions> Element.”

SigningLocation

element **SignatureDefinitionType/SigningLocation**

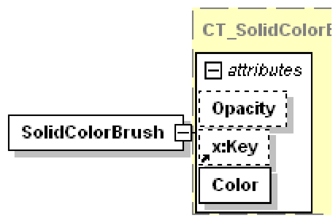
diagram	
annotation	The legal location where the document is signed.

For more information, see §17.2.2.6, “<SigningLocation> Element.”

SolidColorBrush

element **SolidColorBrush**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
Opacity	ST_ZeroOne		1.0		Defines the uniform transparency of the brush fill. Values range from 0 (fully transparent) to 1 (fully opaque), inclusive. Values outside of this range are invalid.
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 [M6.1].
Color	ST_Color	required			Specifies the color for filled elements. An sRGB color value specified as a 6-digit hexadecimal number (#RRGGBB) or an extended color.

annotation

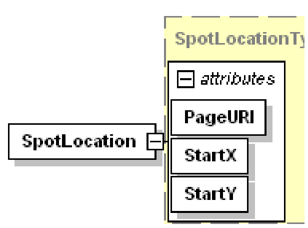
Fills defined geometric regions with a solid color.

For more information, see §13.1, "<SolidColorBrush," on page 126.

SpotLocation

element **SignatureDefinitionType/SpotLocation**

diagram

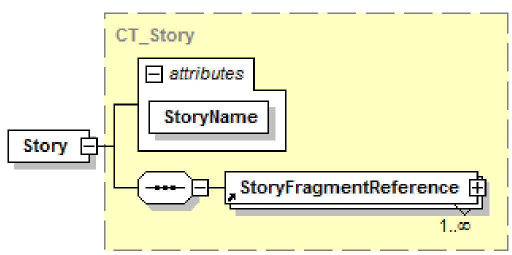


	Name	Type	Use	Default	Fixed	Annotation
attributes	PageURI	xs:anyURI	required			Specifies the page on which the signature spot should be displayed.
	StartX	xs:double	required			Specifies the x coordinate of the origin point (upper-left corner) on the page where the signature spot should be displayed.
	StartY	xs:double	required			Specifies the y coordinate of the origin point (upper-left corner) on the page where the signature spot should be displayed.
annotation	Specifies where a consumer should place a signature spot.					

For more information, see §17.2.2.3, “<SpotLocation> Element.”

Story

element **Story**

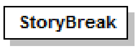
diagram	

	Name	Type	Use	Default	Fixed	Annotation
attributes	StoryName	xs:string	required			The name used by story fragments to identify they belong to this story.
annotation	Defines a single story and where each of its story fragments appear in the XPS Document.					

For more information, see §16.1.1.5, “<Story> Element,” on page 228.

1 **StoryBreak**

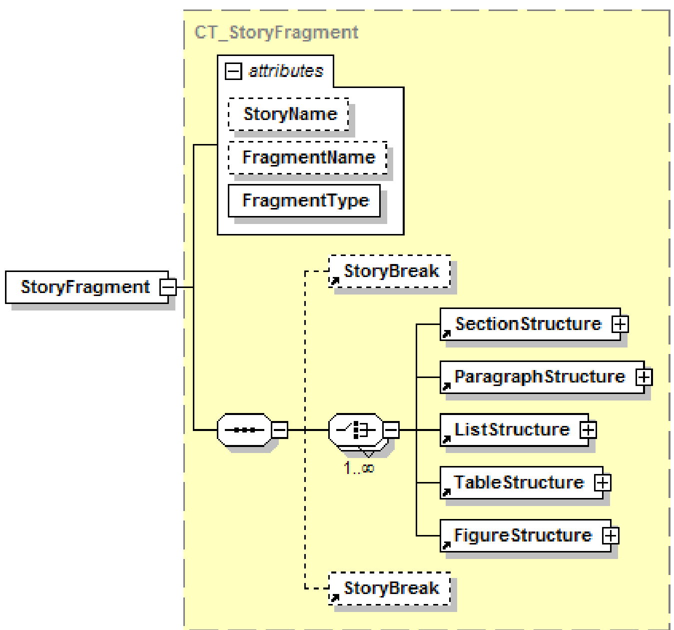
2 element **StoryBreak**

diagram	
annotation	If located at the beginning of a <StoryFragment> definition, indicates that the following markup elements should not be merged with the markup from the previous <StoryFragment>. If located at the end of a <StoryFragment> definition, indicates that the preceding markup elements should not be merged with the subsequent <StoryFragment>.

3 For more information, see §16.1.2.3, "<StoryBreak> Element," on page 239.

4 **StoryFragment**

5 element **StoryFragment**

diagram	

attributes

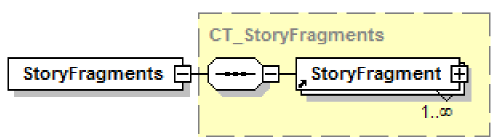
Name	Type	Use	Default	Fixed	Annotation
StoryName	xs:string	optional			Identifies the story that this story fragment belongs to. If omitted, the story fragment is not associated with any story.

	FragmentName	xs:string	optional			Used to uniquely identify the story fragment.
	FragmentType	ST_FragmentType	required			Specifies the type of content included in the story fragment. Valid values are Content, Header, and Footer.
annotation	Specifies the document structural markup that appears on the current page for a single story block.					

For more information, see §16.1.2.2, "<StoryFragment> Element," on page 235.

StoryFragments

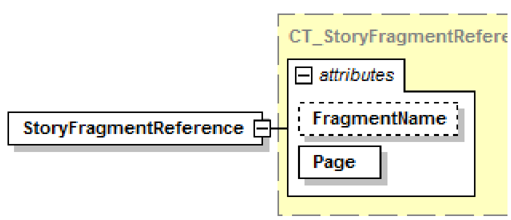
element **StoryFragments**

diagram	
annotation	The root of a StoryFragments part. Contains all story fragments that appear on a specific page.

For more information, see §16.1.2.1, "<StoryFragments> Element," on page 234.

StoryFragmentReference

element **StoryFragmentReference**



	Name	Type	Use	Default	Fixed	Annotation
attributes	FragmentName	xs:string	optional			Used to distingush between multiple story fragments from the same story on a single page.
	Page	ST_IntGEOne	required			Identifies the page number of the document that the story fragment is related to. Page numbers start at 1

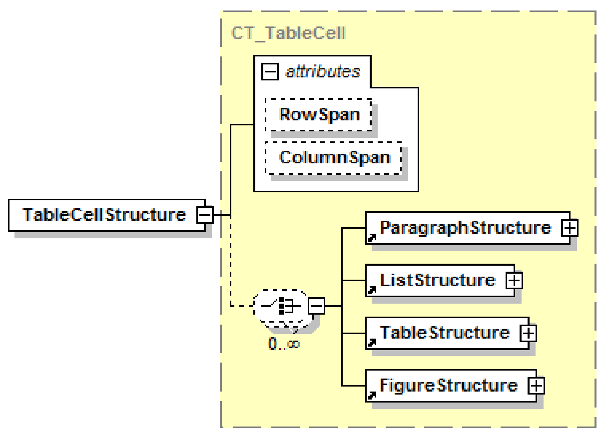
						and correspond to the order of <PageContent> elements in the FixedDocument part.
annotation	Identifies the StoryFragments part where this individual story fragment is defined.					

For more information, see §16.1.1.6, "<StoryFragmentReference> Element," on page 229.

TableCellStructure

element **TableCellStructure**

diagram

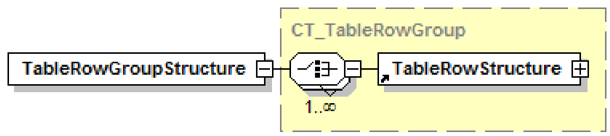


attributes	Name	Type	Use	Default	Fixed	Annotation
	RowSpan	ST_TableSpan	optional	1		Indicates the number of rows this cell spans, or merges into a single cell.
	ColumnSpan	ST_TableSpan	optional	1		Indicates the number of columns this cell spans, or merges into a single cell.
annotation	Contains the elements that occupy a single cell of a table.					

For more information, see §16.1.2.9, "<TableCellStructure> Element," on page 241.

TableRowGroupStructure

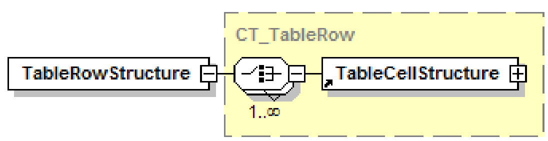
element **TableRowGroupStructure**

diagram	
annotation	Contains the set of table rows that make up a table.

For more information, see §16.1.2.7, “<TableRowGroupStructure> Element,” on page 240.

TableRowStructure

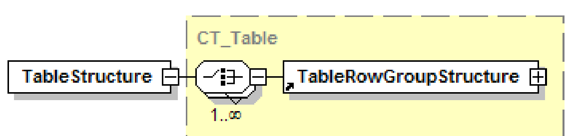
element **TableRowStructure**

diagram	
annotation	Contains the set of table cells that make up a row of a table.

For more information, see §16.1.2.8, “<TableRowStructure> Element,” on page 240.

TableStructure

element **TableStructure**

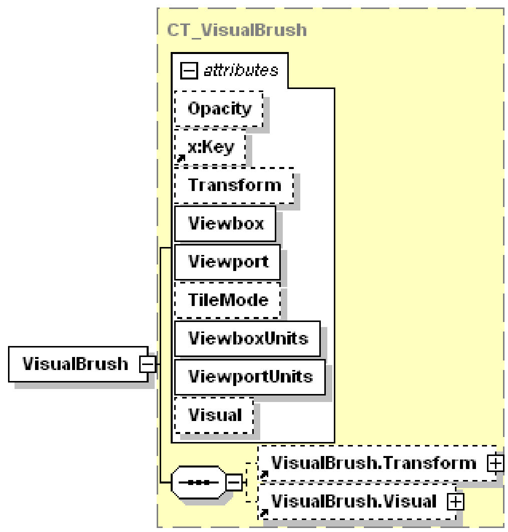
diagram	
annotation	Contains a complete definition of a table in the XPS Document.

For more information, see §16.1.2.6, “<TableStructure> Element,” on page 240.

VisualBrush

element **VisualBrush**

diagram



attributes

Name	Type	Use	Default	Fixed	Annotation
Opacity	ST_ZeroOne		1.0		Defines the uniform transparency of the brush fill. Values range from 0 (fully transparent) to 1 (fully opaque), inclusive. Values outside of this range are invalid.
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 [M6.4].
Transform	ST_RscRefMatrix				Describes the matrix transformation applied to the coordinate space of the brush. The Transform property is concatenated with the current effective render transform to yield an effective render transform local to the brush. The viewport for the brush is transformed using that local

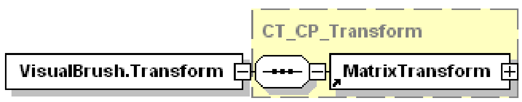
						effective render transform.
	Viewbox	ST_ViewBox	required			Specifies the position and dimensions of the brush's source content. Specifies four comma-separated real numbers (x, y, Width, Height), where width and height are non-negative. The viewbox defines the default coordinate system for the element specified in the <VisualBrush.Visual> property element. The corners of the viewbox are mapped to the corners of the viewport, thereby providing the default clipping and transform for the brush’s source content.
	Viewport	ST_ViewBox	required			Specifies the region in the containing coordinate space of the prime brush tile that is (possibly repeatedly) applied to fill the region to which the brush is applied. Specifies four comma-separated real numbers (x, y, Width, Height), where width and height are non-negative. The alignment of the brush pattern is controlled by adjusting the x and y values.
	TileMode	ST_TileMode		None		Specifies how contents will be tiled in the filled region. Valid values are None, Tile, FlipX, FlipY, and FlipXY.
	ViewboxUnits	ST_ViewUnits	required		Absolute	Specifies the relationship of the viewbox coordinates to the containing coordinate space.
	ViewportUnits	ST_ViewUnits	required		Absolute	Specifies the relationship of the viewport coordinates to the containing coordinate space.
	Visual	ST_RscRef				Specifies resource reference to a <Path>, <Glyphs>, or <Canvas> element defined in a resource dictionary and used to draw the brush’s source content.

annotation	Fills a region with a drawing. The drawing may be specified as either a child of the <VisualBrush> element, or as a resource reference. Drawing content is expressed using <Canvas>, <Path>, and <Glyphs> elements.

For more information, see §13.3, "<VisualBrush> Element," on page 131.

VisualBrush.Transform

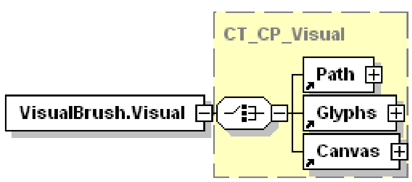
element **VisualBrush.Transform**

diagram	
annotation	Describes the matrix transformation applied to the coordinate space of the brush. The Transform property is concatenated with the current effective render transform to yield an effective render transform local to the brush. The viewport for the brush is transformed using the local effective render transform.

For more information, see §14.4, "Positioning Content," on page 185.

VisualBrush.Visual

element **VisualBrush.Visual**

diagram	
annotation	Specifies a <Path> element, <Glyphs> element, or <Canvas> element used to draw the brush's source contents.

A. Signature Definitions Schema

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema targetNamespace="http://schemas.microsoft.com/xps/2005/06/signature-
definitions" xmlns="http://schemas.microsoft.com/xps/2005/06/signature-
definitions" xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" blockDefault="#all">

  <xs:import namespace="http://www.w3.org/XML/1998/namespace" />

    <xs:complexType name="SignatureDefinitionsType">
        <xs:sequence>
            <xs:element name="SignatureDefinition"
type="SignatureDefinitionType" minOccurs="1" maxOccurs="unbounded">
                </xs:element>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="SpotLocationType">
        <xs:attribute name="PageURI" type="xs:anyURI" use="required">
            </xs:attribute>
        <xs:attribute name="StartX" type="xs:double" use="required">
            </xs:attribute>
        <xs:attribute name="StartY" type="xs:double" use="required">
            </xs:attribute>
    </xs:complexType>

    <xs:complexType name="SignatureDefinitionType">
        <xs:sequence>
            <xs:element name="SpotLocation" type="SpotLocationType"
minOccurs="0">
                </xs:element>
            <xs:element name="Intent" type="xs:string" minOccurs="0">
                </xs:element>
            <xs:element name="SignBy" type="xs:dateTime" minOccurs="0">
                </xs:element>
            <xs:element name="SigningLocation" type="xs:string" minOccurs="0">
                </xs:element>
        </xs:sequence>
        <xs:attribute name="SpotID" type="xs:ID" use="required">
            </xs:attribute>
        <xs:attribute name="SignerName" type="xs:string">
            </xs:attribute>
    <xs:attribute ref="xml:lang">
        </xs:attribute>
    </xs:complexType>

    <xs:element name="SignatureDefinitions" type="SignatureDefinitionsType">
        </xs:element>
</xs:schema>
```

B. XPS Document Schema

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema targetNamespace="http://schemas.microsoft.com/xps/2005/06"
xmlns="http://schemas.microsoft.com/xps/2005/06"
xmlns:mstns="http://tempuri.org/XMLSchema.xsd"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:x="http://schemas.microsoft.com/xps/2005/06/resourcedictionary-key"
elementFormDefault="qualified" blockDefault="#all">

    <xs:import
namespace="http://schemas.microsoft.com/xps/2005/06/resourcedictionary-key" />
    <xs:import namespace="http://www.w3.org/XML/1998/namespace" />

    <!-- Names used for types and groups:

        ST_*        simpleType
        CT_*        complexType
        G_*            group
        AG_*        attributeGroup

    -->

    <!-- Individual real number patterns
    All patterns using numbers now use <whitespace value="collapse">.
    As a result, any whitespace in the pattern can be expressed as:
    mandatory whitespace, one or more:  " "
    optional whitespace, zero or more:  " ?"

    For better readability, each pattern using numbers is also described in a
comment using
    one of the following pattern designators.

    The actual patterns are generated by replacement by the schema publication
process.
    -->
    <!--DEFINE [pint]       "([1-9][0-9]*)" -->
    <!--DEFINE [uint]       "([0-9]+)" -->
    <!--DEFINE [dec]        "(\-?(([0-9]+(\.[0-9]+)?)|(\.[0-9]+)))" -->
    <!--DEFINE [rn]         "((\-|\+)?(([0-9]+(\.[0-9]+)?)|(\.[0-9]+))((e|E)(\-
|\+)?[0-9]+)?)" -->
    <!--DEFINE [prn]        "(\+?(([0-9]+(\.[0-9]+)?)|(\.[0-9]+))((e|E)(\-
|\+)?[0-9]+)?)" -->
    <!--DEFINE [scs]        "( ?, ?)" -->



        <!-- Complex Types -->
    <xs:complexType name="CT_MatrixTransform">
        <xs:attributeGroup ref="AG_MatrixTransform" />
    </xs:complexType>

    <xs:complexType name="CT_SolidColorBrush">
        <xs:attributeGroup ref="AG_Brush" />
        <xs:attributeGroup ref="AG_SolidColorBrush" />
    </xs:complexType>

    <xs:complexType name="CT_ImageBrush">
        <xs:sequence>
```

```
            <xs:element ref="ImageBrush.Transform" minOccurs="0" />
        </xs:sequence>
        <xs:attributeGroup ref="AG_Brush" />
        <xs:attributeGroup ref="AG_TileBrush" />
        <xs:attributeGroup ref="AG_ImageBrush" />
    </xs:complexType>

    <xs:complexType name="CT_VisualBrush">
        <xs:sequence>
            <xs:element ref="VisualBrush.Transform" minOccurs="0" />
            <xs:element ref="VisualBrush.Visual" minOccurs="0" />
        </xs:sequence>
        <xs:attributeGroup ref="AG_Brush" />
        <xs:attributeGroup ref="AG_TileBrush" />
        <xs:attributeGroup ref="AG_VisualBrush" />
    </xs:complexType>

    <xs:complexType name="CT_LinearGradientBrush">
        <xs:sequence>
            <xs:element ref="LinearGradientBrush.Transform" minOccurs="0" />
            <xs:element ref="LinearGradientBrush.GradientStops" />
        </xs:sequence>
        <xs:attributeGroup ref="AG_Brush" />
        <xs:attributeGroup ref="AG_GradientBrush" />
        <xs:attributeGroup ref="AG_LinearGradientBrush" />
    </xs:complexType>

    <xs:complexType name="CT_RadialGradientBrush">
        <xs:sequence>
            <xs:element ref="RadialGradientBrush.Transform" minOccurs="0" />
            <xs:element ref="RadialGradientBrush.GradientStops" />
        </xs:sequence>
        <xs:attributeGroup ref="AG_Brush" />
        <xs:attributeGroup ref="AG_GradientBrush" />
        <xs:attributeGroup ref="AG_RadialGradientBrush" />
    </xs:complexType>

    <xs:complexType name="CT_GradientStop">
        <xs:attributeGroup ref="AG_GradientStop" />
    </xs:complexType>

    <xs:complexType name="CT_PathGeometry">
        <xs:sequence>
            <xs:element ref="PathGeometry.Transform" minOccurs="0" />
            <xs:element ref="PathFigure" minOccurs="0" maxOccurs="unbounded" />
        </xs:sequence>
        <xs:attributeGroup ref="AG_PathGeometry" />
    </xs:complexType>

    <xs:complexType name="CT_Glyphs">
        <xs:sequence>
            <xs:element ref="Glyphs.RenderTransform" minOccurs="0" />
            <xs:element ref="Glyphs.Clip" minOccurs="0" />
            <xs:element ref="Glyphs.OpacityMask" minOccurs="0" />
            <xs:element ref="Glyphs.Fill" minOccurs="0" />
        </xs:sequence>
        <xs:attributeGroup ref="AG_Glyphs" />
    </xs:complexType>

    <xs:complexType name="CT_Path">
        <xs:sequence>
            <xs:element ref="Path.RenderTransform" minOccurs="0" />
```

```
            <xs:element ref="Path.Clip" minOccurs="0" />
            <xs:element ref="Path.OpacityMask" minOccurs="0" />
            <xs:element ref="Path.Fill" minOccurs="0" />
            <xs:element ref="Path.Stroke" minOccurs="0" />
            <xs:element ref="Path.Data" minOccurs="0" />
        </xs:sequence>
        <xs:attributeGroup ref="AG_Path" />
        <xs:attributeGroup ref="AG_AutomationProvider" />
        <xs:attributeGroup ref="AG_SnapsToDevicePixels" />
    </xs:complexType>

    <xs:complexType name="CT_PathFigure">
        <xs:sequence>
            <xs:choice maxOccurs="unbounded">
                <xs:element ref="PolyLineSegment" />
                <xs:element ref="PolyBezierSegment" />
                <xs:element ref="ArcSegment" />
                <xs:element ref="PolyQuadraticBezierSegment" />
            </xs:choice>
        </xs:sequence>
        <xs:attributeGroup ref="AG_PathFigure" />
    </xs:complexType>

    <xs:complexType name="CT_ArcSegment">
        <xs:attributeGroup ref="AG_ArcSegment" />
    </xs:complexType>

    <xs:complexType name="CT_PolyQuadraticBezierSegment">
        <xs:attributeGroup ref="AG_PolyQuadraticBezierSegment" />
    </xs:complexType>

    <xs:complexType name="CT_PolyLineSegment">
        <xs:attributeGroup ref="AG_PolyLineSegment" />
    </xs:complexType>

    <xs:complexType name="CT_PolyBezierSegment">
        <xs:attributeGroup ref="AG_PolyBezierSegment" />
    </xs:complexType>

    <xs:complexType name="CT_Canvas">
        <xs:sequence>
            <xs:element ref="Canvas.Resources" minOccurs="0" />
            <xs:element ref="Canvas.RenderTransform" minOccurs="0" />
            <xs:element ref="Canvas.Clip" minOccurs="0" />
            <xs:element ref="Canvas.OpacityMask" minOccurs="0" />
            <xs:choice minOccurs="0" maxOccurs="unbounded">
                <xs:element ref="Path" />
                <xs:element ref="Glyphs" />
                <xs:element ref="Canvas" />
            </xs:choice>
        </xs:sequence>
        <xs:attributeGroup ref="AG_Canvas" />
        <xs:attributeGroup ref="AG_AutomationProvider" />
    </xs:complexType>

    <xs:complexType name="CT_ResourceDictionary">
        <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:element ref="ImageBrush" />
            <xs:element ref="LinearGradientBrush" />
            <xs:element ref="RadialGradientBrush" />
            <xs:element ref="VisualBrush" />
            <xs:element ref="SolidColorBrush" />
            <xs:element ref="MatrixTransform" />
```

```
            <xs:element ref="PathGeometry" />
            <xs:element ref="Path" />
            <xs:element ref="Glyphs" />
            <xs:element ref="Canvas" />
        </xs:choice>
        <xs:attributeGroup ref="AG_ResourceDictionary" />
    </xs:complexType>

    <xs:complexType name="CT_FixedPage">
        <xs:sequence>
            <xs:element ref="FixedPage.Resources" minOccurs="0" />
            <xs:choice minOccurs="0" maxOccurs="unbounded">
                <xs:element ref="Path" />
                <xs:element ref="Glyphs" />
                <xs:element ref="Canvas" />
            </xs:choice>
        </xs:sequence>
        <xs:attributeGroup ref="AG_FixedPage" />
    </xs:complexType>

    <xs:complexType name="CT_FixedDocument">
        <xs:sequence>
            <xs:element ref="PageContent" maxOccurs="unbounded" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="CT_PageContent">
        <xs:sequence>
            <xs:element ref="PageContent.LinkTargets" minOccurs="0" />
        </xs:sequence>
        <xs:attributeGroup ref="AG_PageContent" />
    </xs:complexType>

    <xs:complexType name="CT_FixedDocumentSequence">
        <xs:sequence>
            <xs:element ref="DocumentReference" maxOccurs="unbounded" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="CT_DocumentReference">
        <xs:attributeGroup ref="AG_DocumentReference" />
    </xs:complexType>

    <xs:complexType name="CT_LinkTarget">
        <xs:attributeGroup ref="AG_LinkTarget" />
    </xs:complexType>

    <xs:complexType name="CT_CP_LinkTargets">
        <xs:sequence>
            <xs:element ref="LinkTarget" maxOccurs="unbounded" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="CT_CP_Transform">
        <xs:sequence>
            <xs:element ref="MatrixTransform" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="CT_CP_Visual">
        <xs:choice>
            <xs:element ref="Path" />
```

```
            <xs:element ref="Glyphs" />
            <xs:element ref="Canvas" />
        </xs:choice>
    </xs:complexType>

    <xs:complexType name="CT_CP_GradientStops">
        <xs:sequence>
            <xs:element ref="GradientStop" minOccurs="2" maxOccurs="unbounded"
/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="CT_CP_Geometry">
        <xs:sequence>
        <xs:element ref="PathGeometry" />
      </xs:sequence>
    </xs:complexType>

    <xs:complexType name="CT_CP_Brush">
        <xs:choice>
            <xs:element ref="ImageBrush" />
            <xs:element ref="LinearGradientBrush" />
            <xs:element ref="RadialGradientBrush" />
            <xs:element ref="SolidColorBrush" />
            <xs:element ref="VisualBrush" />
        </xs:choice>
    </xs:complexType>

    <xs:complexType name="CT_CP_Resources">
        <xs:sequence minOccurs="0">
            <xs:element ref="ResourceDictionary" />
        </xs:sequence>
    </xs:complexType>

    <!-- Root elements -->
    <xs:element name="MatrixTransform" type="CT_MatrixTransform">
        </xs:element>

    <xs:element name="SolidColorBrush" type="CT_SolidColorBrush">
        </xs:element>

    <xs:element name="ImageBrush" type="CT_ImageBrush">
        </xs:element>

    <xs:element name="VisualBrush" type="CT_VisualBrush">
        </xs:element>

    <xs:element name="LinearGradientBrush" type="CT_LinearGradientBrush">
        </xs:element>

    <xs:element name="RadialGradientBrush" type="CT_RadialGradientBrush">
        </xs:element>

    <xs:element name="Glyphs" type="CT_Glyphs">
        </xs:element>

    <xs:element name="Path" type="CT_Path">
        </xs:element>

    <xs:element name="Canvas" type="CT_Canvas">
        </xs:element>

    <xs:element name="GradientStop" type="CT_GradientStop">
```

```
        </xs:element>

    <xs:element name="ResourceDictionary" type="CT_ResourceDictionary">
        </xs:element>

    <xs:element name="PathGeometry" type="CT_PathGeometry">
        </xs:element>

    <xs:element name="PathFigure" type="CT_PathFigure">
        </xs:element>

    <xs:element name="PolyLineSegment" type="CT_PolyLineSegment">
        </xs:element>

    <xs:element name="ArcSegment" type="CT_ArcSegment">
        </xs:element>

    <xs:element name="PolyBezierSegment" type="CT_PolyBezierSegment">
        </xs:element>

    <xs:element name="PolyQuadraticBezierSegment"
type="CT_PolyQuadraticBezierSegment">
        </xs:element>

    <xs:element name="FixedPage" type="CT_FixedPage">
        </xs:element>

    <xs:element name="FixedDocument" type="CT_FixedDocument">
        </xs:element>

    <xs:element name="PageContent" type="CT_PageContent">
        </xs:element>

    <xs:element name="FixedDocumentSequence" type="CT_FixedDocumentSequence">
        </xs:element>

    <xs:element name="DocumentReference" type="CT_DocumentReference">
        </xs:element>

    <xs:element name="LinkTarget" type="CT_LinkTarget">
        </xs:element>

    <xs:element name="PageContent.LinkTargets" type="CT_CP_LinkTargets">
        </xs:element>

    <xs:element name="ImageBrush.Transform" type="CT_CP_Transform">
        </xs:element>

    <xs:element name="VisualBrush.Transform" type="CT_CP_Transform">
        </xs:element>

    <xs:element name="LinearGradientBrush.Transform" type="CT_CP_Transform">
        </xs:element>

    <xs:element name="RadialGradientBrush.Transform" type="CT_CP_Transform">
        </xs:element>

    <xs:element name="PathGeometry.Transform" type="CT_CP_Transform">
        </xs:element>

    <xs:element name="Glyphs.RenderTransform" type="CT_CP_Transform">
        </xs:element>
```

```

    <xs:element name="Path.RenderTransform" type="CT_CP_Transform">
        </xs:element>

    <xs:element name="Canvas.RenderTransform" type="CT_CP_Transform">
        </xs:element>

    <xs:element name="VisualBrush.Visual" type="CT_CP_Visual">
        </xs:element>

    <xs:element name="LinearGradientBrush.GradientStops"
type="CT_CP_GradientStops">
        </xs:element>

    <xs:element name="RadialGradientBrush.GradientStops"
type="CT_CP_GradientStops">
        </xs:element>

    <xs:element name="Glyphs.Clip" type="CT_CP_Geometry">
        </xs:element>

    <xs:element name="Path.Clip" type="CT_CP_Geometry">
        </xs:element>

    <xs:element name="Canvas.Clip" type="CT_CP_Geometry">
        </xs:element>

    <xs:element name="Glyphs.OpacityMask" type="CT_CP_Brush">
        </xs:element>

    <xs:element name="Path.OpacityMask" type="CT_CP_Brush">
        </xs:element>

    <xs:element name="Canvas.OpacityMask" type="CT_CP_Brush">
        </xs:element>

    <xs:element name="Glyphs.Fill" type="CT_CP_Brush">
        </xs:element>

    <xs:element name="Path.Fill" type="CT_CP_Brush">
        </xs:element>

    <xs:element name="Path.Data" type="CT_CP_Geometry">
        </xs:element>

    <xs:element name="Path.Stroke" type="CT_CP_Brush">
        </xs:element>

    <xs:element name="Canvas.Resources" type="CT_CP_Resources">
        </xs:element>

    <xs:element name="FixedPage.Resources" type="CT_CP_Resources">
        </xs:element>

    <xs:attributeGroup name="AG_GradientStop">
        <xs:attribute name="Color" type="ST_Color" use="required">
            </xs:attribute>
        <xs:attribute name="Offset" type="ST_Double" use="required">
            </xs:attribute>
    </xs:attributeGroup>

    <xs:attributeGroup name="AG_Brush">
        <xs:attribute name="Opacity" type="ST_ZeroOne" default="1.0">
```

```
            </xs:attribute>
        <xs:attribute ref="x:Key" />
    </xs:attributeGroup>

    <xs:attributeGroup name="AG_TileBrush">
        <xs:attribute name="Transform" type="ST_RscRefMatrix">
            </xs:attribute>
        <xs:attribute name="Viewbox" type="ST_ViewBox" use="required">
            </xs:attribute>
        <xs:attribute name="Viewport" type="ST_ViewBox" use="required">
            </xs:attribute>
        <xs:attribute name="TileMode" type="ST_TileMode" default="None">
            </xs:attribute>
        <xs:attribute name="ViewboxUnits" type="ST_ViewUnits" use="required"
fixed="Absolute">
            </xs:attribute>
        <xs:attribute name="ViewportUnits" type="ST_ViewUnits" use="required"
fixed="Absolute">
            </xs:attribute>
    </xs:attributeGroup>

    <xs:attributeGroup name="AG_VisualBrush">
        <xs:attribute name="Visual" type="ST_RscRef">
            </xs:attribute>
    </xs:attributeGroup>

    <xs:attributeGroup name="AG_GradientBrush">
        <xs:attribute name="ColorInterpolationMode" type="ST_ClrIntMode"
default="SRgbLinearInterpolation">
            </xs:attribute>
        <xs:attribute name="SpreadMethod" type="ST_SpreadMethod" default="Pad">
            </xs:attribute>
        <xs:attribute name="MappingMode" type="ST_MappingMode" use="required"
fixed="Absolute">
            </xs:attribute>
    </xs:attributeGroup>

    <xs:attributeGroup name="AG_SolidColorBrush">
        <xs:attribute name="Color" type="ST_Color" use="required">
            </xs:attribute>
    </xs:attributeGroup>

    <xs:attributeGroup name="AG_ImageBrush">
        <xs:attribute name="ImageSource" type="ST_UriCtxBmp" use="required">
            </xs:attribute>
    </xs:attributeGroup>

    <xs:attributeGroup name="AG_LinearGradientBrush">
        <xs:attribute name="Transform" type="ST_RscRefMatrix">
            </xs:attribute>
        <xs:attribute name="StartPoint" type="ST_Point" use="required">
            </xs:attribute>
        <xs:attribute name="EndPoint" type="ST_Point" use="required">
            </xs:attribute>
    </xs:attributeGroup>

    <xs:attributeGroup name="AG_RadialGradientBrush">
        <xs:attribute name="Transform" type="ST_RscRefMatrix">
            </xs:attribute>
        <xs:attribute name="Center" type="ST_Point" use="required">
            </xs:attribute>
        <xs:attribute name="GradientOrigin" type="ST_Point" use="required">
```

```
            </xs:attribute>
        <xs:attribute name="RadiusX" type="ST_GEZero" use="required">
            </xs:attribute>
        <xs:attribute name="RadiusY" type="ST_GEZero" use="required">
            </xs:attribute>
    </xs:attributeGroup>

    <xs:attributeGroup name="AG_PathGeometry">
        <xs:attribute name="Figures" type="ST_AbbrGeom">
            </xs:attribute>
        <xs:attribute name="FillRule" type="ST_FillRule" default="EvenOdd">
            </xs:attribute>
        <xs:attribute name="Transform" type="ST_RscRefMatrix">
            </xs:attribute>
        <xs:attribute ref="x:Key" />
    </xs:attributeGroup>

    <xs:attributeGroup name="AG_ResourceDictionary">
        <xs:attribute name="Source" type="xs:anyURI">
            </xs:attribute>
    </xs:attributeGroup>

    <xs:attributeGroup name="AG_PolyLineSegment">
        <xs:attribute name="Points" type="ST_Points" use="required">
            </xs:attribute>
        <xs:attribute name="IsStroked" type="ST_Boolean" default="true">
            </xs:attribute>
    </xs:attributeGroup>

    <xs:attributeGroup name="AG_ArcSegment">
        <xs:attribute name="Point" type="ST_Point" use="required">
            </xs:attribute>
        <xs:attribute name="Size" type="ST_PointGE0" use="required">
            </xs:attribute>
        <xs:attribute name="RotationAngle" type="ST_Double" use="required">
            </xs:attribute>
        <xs:attribute name="IsLargeArc" type="ST_Boolean" use="required">
            </xs:attribute>
        <xs:attribute name="SweepDirection" type="ST_SweepDirection"
use="required">
            </xs:attribute>
        <xs:attribute name="IsStroked" type="ST_Boolean" default="true">
            </xs:attribute>
    </xs:attributeGroup>

    <xs:attributeGroup name="AG_PolyBezierSegment">
        <xs:attribute name="Points" type="ST_Points" use="required">
            </xs:attribute>
        <xs:attribute name="IsStroked" type="ST_Boolean" default="true">
            </xs:attribute>
    </xs:attributeGroup>

    <xs:attributeGroup name="AG_PolyQuadraticBezierSegment">
        <xs:attribute name="Points" type="ST_Points" use="required">
            </xs:attribute>
        <xs:attribute name="IsStroked" type="ST_Boolean" default="true">
            </xs:attribute>
    </xs:attributeGroup>

    <xs:attributeGroup name="AG_Glyphs">
        <xs:attribute name="BidiLevel" default="0">
            <xs:simpleType>
              <xs:restriction base="xs:integer">
```

```
            <xs:minInclusive value="0" />
            <xs:maxInclusive value="61" />
          </xs:restriction>
        </xs:simpleType>
        </xs:attribute>
    <xs:attribute name="CaretStops" type="ST_CaretStops">
        </xs:attribute>
    <xs:attribute name="DeviceFontName" type="ST_UnicodeString">
        </xs:attribute>
    <xs:attribute name="Fill" type="ST_RscRefColor">
        </xs:attribute>
    <xs:attribute name="FontRenderingEmSize" type="ST_GEZero" use="required">
        </xs:attribute>
    <xs:attribute name="FontUri" type="xs:anyURI" use="required">
        </xs:attribute>
    <xs:attribute name="OriginX" type="ST_Double" use="required">
        </xs:attribute>
    <xs:attribute name="OriginY" type="ST_Double" use="required">
        </xs:attribute>
    <xs:attribute name="IsSideways" type="ST_Boolean" default="false">
        </xs:attribute>
    <xs:attribute name="Indices" type="ST_Indices">
        </xs:attribute>
    <xs:attribute name="UnicodeString" type="ST_UnicodeString">
        </xs:attribute>
    <xs:attribute name="StyleSimulations" type="ST_StyleSimulations" default="None">
        </xs:attribute>
    <xs:attribute name="RenderTransform" type="ST_RscRefMatrix">
        </xs:attribute>
    <xs:attribute name="Clip" type="ST_RscRefAbbrGeomF">
        </xs:attribute>
    <xs:attribute name="Opacity" type="ST_ZeroOne" default="1.0">
        </xs:attribute>
    <xs:attribute name="OpacityMask" type="ST_RscRef">
        </xs:attribute>
    <xs:attribute name="Name" type="ST_Name">
        </xs:attribute>
    <xs:attribute name="FixedPage.NavigateUri" type="xs:anyURI">
        </xs:attribute>
    <xs:attribute ref="xml:lang">
        </xs:attribute>
    <xs:attribute ref="x:Key" />
</xs:attributeGroup>

<xs:attributeGroup name="AG_Path">
    <xs:attribute name="Data" type="ST_RscRefAbbrGeomF">
        </xs:attribute>
    <xs:attribute name="Fill" type="ST_RscRefColor">
        </xs:attribute>
    <xs:attribute name="RenderTransform" type="ST_RscRefMatrix">
        </xs:attribute>
    <xs:attribute name="Clip" type="ST_RscRefAbbrGeomF">
        </xs:attribute>
    <xs:attribute name="Opacity" type="ST_ZeroOne" default="1.0">
        </xs:attribute>
    <xs:attribute name="OpacityMask" type="ST_RscRef">
        </xs:attribute>
    <xs:attribute name="Stroke" type="ST_RscRefColor">
        </xs:attribute>
    <xs:attribute name="StrokeDashArray" type="ST_EvenArrayPos">
```

```
            </xs:attribute>
        <xs:attribute name="StrokeDashCap" type="ST_DashCap" default="Flat">
            </xs:attribute>
        <xs:attribute name="StrokeDashOffset" type="ST_Double" default="0.0">
            </xs:attribute>
        <xs:attribute name="StrokeEndLineCap" type="ST_LineCap" default="Flat">
            </xs:attribute>
        <xs:attribute name="StrokeStartLineCap" type="ST_LineCap"
default="Flat">
            </xs:attribute>
        <xs:attribute name="StrokeLineJoin" type="ST_LineJoin" default="Miter">
            </xs:attribute>
        <xs:attribute name="StrokeMiterLimit" type="ST_GEOne" default="10.0">
            </xs:attribute>
        <xs:attribute name="StrokeThickness" type="ST_GEZero" default="1.0">
            </xs:attribute>
        <xs:attribute name="Name" type="ST_Name">
            </xs:attribute>
        <xs:attribute name="FixedPage.NavigateUri" type="xs:anyURI">
            </xs:attribute>
        <xs:attribute ref="xml:lang">
            </xs:attribute>
        <xs:attribute ref="x:Key" />
    </xs:attributeGroup>

    <xs:attributeGroup name="AG_PathFigure">
        <xs:attribute name="IsClosed" type="ST_Boolean" default="false">
            </xs:attribute>
        <xs:attribute name="StartPoint" type="ST_Point" use="required">
            </xs:attribute>
        <xs:attribute name="IsFilled" type="ST_Boolean" default="true">
            </xs:attribute>
    </xs:attributeGroup>

    <xs:attributeGroup name="AG_Canvas">
        <xs:attribute name="RenderTransform" type="ST_RscRefMatrix">
            </xs:attribute>
        <xs:attribute name="Clip" type="ST_RscRefAbbrGeomF">
            </xs:attribute>
        <xs:attribute name="Opacity" type="ST_ZeroOne" default="1.0">
            </xs:attribute>
        <xs:attribute name="OpacityMask" type="ST_RscRef">
            </xs:attribute>
        <xs:attribute name="Name" type="ST_Name">
            </xs:attribute>
        <xs:attribute name="RenderOptions.EdgeMode" type="ST_EdgeMode">
            </xs:attribute>
        <xs:attribute name="FixedPage.NavigateUri" type="xs:anyURI">
            </xs:attribute>
        <xs:attribute ref="xml:lang">
            </xs:attribute>
        <xs:attribute ref="x:Key" />
    </xs:attributeGroup>

    <xs:attributeGroup name="AG_PageContent">
        <xs:attribute name="Source" type="xs:anyURI" use="required">
            </xs:attribute>
        <xs:attribute name="Width" type="ST_GEOne">
            </xs:attribute>
        <xs:attribute name="Height" type="ST_GEOne">
            </xs:attribute>
    </xs:attributeGroup>
```

```
    <xs:attributeGroup name="AG_LinkTarget">
        <xs:attribute name="Name" type="ST_NUName" use="required">
            </xs:attribute>
    </xs:attributeGroup>

    <xs:attributeGroup name="AG_DocumentReference">
        <xs:attribute name="Source" type="xs:anyURI" use="required">
            </xs:attribute>
    </xs:attributeGroup>

    <xs:attributeGroup name="AG_MatrixTransform">
        <xs:attribute name="Matrix" type="ST_Matrix" use="required">
            </xs:attribute>
        <xs:attribute ref="x:Key" />
    </xs:attributeGroup>

    <xs:attributeGroup name="AG_FixedPage">
        <xs:attribute name="Width" type="ST_GEOne" use="required">
            </xs:attribute>
        <xs:attribute name="Height" type="ST_GEOne" use="required">
            </xs:attribute>
        <xs:attribute name="ContentBox" type="ST_ContentBox">
            </xs:attribute>
        <xs:attribute name="BleedBox" type="ST_BleedBox">
            </xs:attribute>
        <xs:attribute ref="xml:lang" use="required">
            </xs:attribute>
        <xs:attribute name="Name" type="ST_Name">
            </xs:attribute>
    </xs:attributeGroup>

    <xs:attributeGroup name="AG_AutomationProvider">
        <xs:attribute name="AutomationProperties.Name" type="xs:string">
            </xs:attribute>
        <xs:attribute name="AutomationProperties.HelpText" type="xs:string">
            </xs:attribute>
    </xs:attributeGroup>

    <xs:attributeGroup name="AG_SnapsToDevicePixels">
        <xs:attribute name="SnapsToDevicePixels" type="ST_Boolean">
            </xs:attribute>
    </xs:attributeGroup>

    <!-- Simple data types -->
    <!-- A unique Name (ID with pattern restriction according to XPS spec) -->
    <xs:simpleType name="ST_Name">
        <xs:restriction base="xs:ID">
            <xs:pattern
value="(\p{Lu}|\p{Ll}|\p{Lt}|\p{Lo}|\p{Nl}|_)(\p{Lu}|\p{Ll}|\p{Lt}|\p{Lo}|\p{Nl
}|\p{Mn}|\p{Mc}|\p{Nd}|_)*" />
        </xs:restriction>
    </xs:simpleType>

    <!-- A non-unique Name (ID with pattern restriction according to XPS spec)
-->
    <xs:simpleType name="ST_NUName">
        <xs:restriction base="xs:string">
            <xs:pattern
value="(\p{Lu}|\p{Ll}|\p{Lt}|\p{Lo}|\p{Nl}|_)(\p{Lu}|\p{Ll}|\p{Lt}|\p{Lo}|\p{Nl
}|\p{Mn}|\p{Mc}|\p{Nd}|_)*" />
        </xs:restriction>
    </xs:simpleType>
```

```
    <!-- Boolean with true and false only (no 0 or 1) -->
    <xs:simpleType name="ST_Boolean">
        <xs:restriction base="xs:boolean">
            <xs:pattern value="true|false" />
        </xs:restriction>
    </xs:simpleType>

    <!-- real number from 0.0 to 1.0 inclusive -->
    <xs:simpleType name="ST_ZeroOne">
        <xs:restriction base="ST_Double">
            <xs:minInclusive value="0.0" />
            <xs:maxInclusive value="1.0" />
        </xs:restriction>
    </xs:simpleType>

    <!-- positive real number -->
    <xs:simpleType name="ST_GEZero">
        <xs:restriction base="ST_Double">
            <xs:minInclusive value="0.0" />
        </xs:restriction>
    </xs:simpleType>

    <!-- positive real number, equal or greater than one -->
    <xs:simpleType name="ST_GEOne">
        <xs:restriction base="ST_Double">
            <xs:minInclusive value="1.0" />
        </xs:restriction>
    </xs:simpleType>

    <!-- Double -->
    <xs:simpleType name="ST_Double">
        <xs:restriction base="xs:double">
            <xs:whiteSpace value="collapse" />
<!--
            <xs:pattern value="[rn]"/>
-->
            <xs:pattern value="((\-|\+)?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?)" />
        </xs:restriction>
    </xs:simpleType>

    <!-- Point: 2 numbers, separated by , and arbitrary whitespace -->
    <xs:simpleType name="ST_Point">
        <xs:restriction base="xs:string">
            <xs:whiteSpace value="collapse" />
<!--
            <xs:pattern value="[rn][scs][rn]"/>
-->
            <xs:pattern value="((\-|\+)?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)((\-|\+)?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?)" />
        </xs:restriction>
    </xs:simpleType>

    <!-- PointGE0: 2 non-negative numbers, separated by , and arbitrary
whitespace -->
    <xs:simpleType name="ST_PointGE0">
        <xs:restriction base="xs:string">
            <xs:whiteSpace value="collapse" />
<!--
            <xs:pattern value="[prn][scs][prn]"/>
-->
```

```
            <xs:pattern value="(\+?(([0-9]+(\.[0-9]+)?)|(\.[0-9]+))((e|E)(\-
|\+)?[0-9]+)?)( ?, ?)(\+?(([0-9]+(\.[0-9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-
9]+)?)" />
        </xs:restriction>
    </xs:simpleType>

    <!-- Points: List of ST_Point, separated by arbitrary whitespace -->
    <xs:simpleType name="ST_Points">
        <xs:restriction base="xs:string">
            <xs:whiteSpace value="collapse" />
<!--
            <xs:pattern value="[rn][scs][rn]( [rn][scs][rn])*"/>
-->
            <xs:pattern value="((\-|\+)?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)((\-|\+)?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?)( ((\-|\+)?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)((\-|\+)?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?))*" />
        </xs:restriction>
    </xs:simpleType>

    <!-- EvenArray: List with even number of entries of non-negative numbers.
-->
    <xs:simpleType name="ST_EvenArrayPos">
        <xs:restriction base="xs:string">
            <xs:whiteSpace value="collapse" />
<!--
            <xs:pattern value="[prn] [prn]( [prn] [prn])*"/>
-->
            <xs:pattern value="(\+?(([0-9]+(\.[0-9]+)?)|(\.[0-9]+))((e|E)(\-
|\+)?[0-9]+)?) (\+?(([0-9]+(\.[0-9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?)(
(\+?(([0-9]+(\.[0-9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?) (\+?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?))*" />
        </xs:restriction>
    </xs:simpleType>

    <!-- Array: List of numbers.  -->
    <xs:simpleType name="ST_Array">
        <xs:restriction base="xs:string">
            <xs:whiteSpace value="collapse" />
<!--
            <xs:pattern value="([rn] ?)*"/>
-->
            <xs:pattern value="(((\-|\+)?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?) ?)*" />
        </xs:restriction>
    </xs:simpleType>

    <!-- ViewBox: 4 numbers, separated by , and arbitrary whitespace. Second
number pair must be non-negative -->
    <xs:simpleType name="ST_ViewBox">
        <xs:restriction base="xs:string">
            <xs:whiteSpace value="collapse" />
<!--
            <xs:pattern value="[rn][scs][rn][scs][prn][scs][prn]"/>
-->
            <xs:pattern value="((\-|\+)?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)((\-|\+)?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)(\+?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)(\+?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?)" />
        </xs:restriction>
```

```
    </xs:simpleType>

    <!-- ContentBox: 4 non-negative numbers, separated by commas and arbitrary
whitespace -->
    <xs:simpleType name="ST_ContentBox">
        <xs:restriction base="xs:string">
            <xs:whiteSpace value="collapse" />
<!--
            <xs:pattern value="[prn][scs][prn][scs][prn][scs][prn]"/>
-->
            <xs:pattern value="(\+?(([0-9]+(\.[0-9]+)?)|(\.[0-9]+))((e|E)(\-
|\+)?[0-9]+)?)( ?, ?)(\+?(([0-9]+(\.[0-9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-
9]+)?)( ?, ?)(\+?(([0-9]+(\.[0-9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?)( ?,
?)(\+?(([0-9]+(\.[0-9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?)" />
        </xs:restriction>
    </xs:simpleType>

    <!-- BleedBox: 4 numbers, separated by , and arbitrary whitespace. Second
number pair must be non-negative -->
        <xs:simpleType name="ST_BleedBox">
        <xs:restriction base="xs:string">
            <xs:whiteSpace value="collapse" />
<!--
            <xs:pattern value="[rn][scs][rn][scs][prn][scs][prn]"/>
-->
            <xs:pattern value="((\-|\+)?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)((\-|\+)?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)(\+?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)(\+?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?)" />
        </xs:restriction>
    </xs:simpleType>

    <!-- Bare Matrix form: 6 numbers separated by , and arbitrary whitespace --
>
    <xs:simpleType name="ST_Matrix">
        <xs:restriction base="xs:string">
            <xs:whiteSpace value="collapse" />
<!--
            <xs:pattern
value="[rn][scs][rn][scs][rn][scs][rn][scs][rn][scs][rn]"/>
-->
            <xs:pattern value="((\-|\+)?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)((\-|\+)?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)((\-|\+)?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)((\-|\+)?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)((\-|\+)?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)((\-|\+)?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?)" />
        </xs:restriction>
    </xs:simpleType>

    <!-- Color: 6 or 8 hex digits -->
    <xs:simpleType name="ST_Color">
        <xs:restriction base="xs:string">
            <!-- The pattern restriction does not check for scRGB gamut -->
            <!-- The pattern restriction does not check for color profile URI
validity -->
            <xs:whiteSpace value="collapse" />
<!--
            <xs:pattern value="(#([0-9a-fA-F]{2})?[0-9a-fA-F]{6})|\
                                (sc# ?[dec][scs][dec][scs][dec]([scs][dec])?)|\
```

```
                                (ContextColor +[\S]+
?[dec]([scs][dec]){3,8})"/>
-->
            <xs:pattern value="(#([0-9a-fA-F]{2})?[0-9a-fA-F]{6})|(sc# ?(\-
?(([0-9]+(\.[0-9]+)?)|(\.[0-9]+)))( ?, ?)(\-?(([0-9]+(\.[0-9]+)?)|(\.[0-9]+)))(
?, ?)(\-?(([0-9]+(\.[0-9]+)?)|(\.[0-9]+)))(( ?, ?)(\-?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))))?)|(ContextColor +[\S]+ ?(\-?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+)))(( ?, ?)(\-?(([0-9]+(\.[0-9]+)?)|(\.[0-9]+)))){3,8})" />
        </xs:restriction>
    </xs:simpleType>

    <!-- Indices grammar for Glyphs.CaretStops -->
    <xs:simpleType name="ST_CaretStops">
        <xs:restriction base="xs:string">
            <xs:whiteSpace value="collapse" />
            <xs:pattern value="[0-9A-Fa-f]*" />
        </xs:restriction>
    </xs:simpleType>

    <!-- Indices grammar for Glyphs.Indices -->
    <xs:simpleType name="ST_Indices">
        <xs:restriction base="xs:string">
            <xs:whiteSpace value="collapse" />
<!--
            <xs:pattern value="(\
                                ((\([pint](:[pint])?\))?[uint])?\
                                (,[prn]?(,[rn]?(,[rn])?)?)?\
                                )\
                                (;\
                                    ((\([pint](:[pint])?\))?[uint])?\
                                    (,[prn]?(,[rn]?(,[rn])?)?)?\
                                )*" />
-->
            <xs:pattern value="(((\(([1-9][0-9]*)(:([1-9][0-9]*))?\))?([0-
9]+))?(,(\+?(([0-9]+(\.[0-9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?)?(,((\-
|\+)?(([0-9]+(\.[0-9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?)?(,((\-|\+)?(([0-
9]+(\.[0-9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?))?)?)?)(;((\(([1-9][0-
9]*)(:([1-9][0-9]*))?\))?([0-9]+))?(,(\+?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?)?(,((\-|\+)?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?)?(,((\-|\+)?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?))?)?)?)*" />
        </xs:restriction>
    </xs:simpleType>

    <!-- UnicodeString grammar -->
    <xs:simpleType name="ST_UnicodeString">
        <xs:restriction base="xs:string">
            <xs:pattern value="(([^\{]|(\{\})).*)?" />
        </xs:restriction>
    </xs:simpleType>

    <!-- Abbreviated Geometry grammar for Path.Data , clip and Geometries -->
    <xs:simpleType name="ST_AbbrGeomF">
        <xs:restriction base="xs:string">
            <xs:whiteSpace value="collapse" />
<!--
            <xs:pattern value="(F ?(0|1))?\
                                ( ?(M|m)( ?[rn][scs][rn]))\
                                (\
                                    ( ?(M|m)( ?[rn][scs][rn]))|\
                                    ( ?(L|l)( ?[rn][scs][rn])(
[rn][scs][rn])*)|\
```

```
                                    ( ?(H|h|V|v)( ?[rn])( [rn])*)|\
                                    ( ?(Q|q|S|s)( ?[rn][scs][rn]
[rn][scs][rn])(( [rn][scs][rn]){2})*)|\
                                    ( ?(C|c)( ?[rn][scs][rn](
[rn][scs][rn]){2})(( [rn][scs][rn]){3})*)|\
                                    ( ?(A|a)( ?[rn][scs][rn] [rn] [0-1] [0-1]
[rn][scs][rn])\
                                                    ( [rn][scs][rn] [rn] [0-1]
[0-1] [rn][scs][rn])*)|\
                                    ( ?(Z|z))\
                                )*"/>
-->
            <xs:pattern value="(F ?(0|1))?( ?(M|m)( ?((\-|\+)?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)((\-|\+)?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?)))(( ?(M|m)( ?((\-|\+)?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)((\-|\+)?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?)))|( ?(L|l)( ?((\-|\+)?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)((\-|\+)?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?))( ((\-|\+)?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)((\-|\+)?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?))*)|( ?(H|h|V|v)( ?((\-|\+)?(([0-
9]+(\.[0-9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?))( ((\-|\+)?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?))*)|( ?(Q|q|S|s)( ?((\-|\+)?(([0-
9]+(\.[0-9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)((\-|\+)?(([0-
9]+(\.[0-9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?) ((\-|\+)?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)((\-|\+)?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?))(( ((\-|\+)?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)((\-|\+)?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?)){2})*)|( ?(C|c)( ?((\-|\+)?(([0-
9]+(\.[0-9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)((\-|\+)?(([0-
9]+(\.[0-9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?)( ((\-|\+)?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)((\-|\+)?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?)){2})(( ((\-|\+)?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)((\-|\+)?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?)){3})*)|( ?(A|a)( ?((\-|\+)?(([0-
9]+(\.[0-9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)((\-|\+)?(([0-
9]+(\.[0-9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?) ((\-|\+)?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?) [0-1] [0-1] ((\-|\+)?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)((\-|\+)?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?))( ((\-|\+)?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)((\-|\+)?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?) ((\-|\+)?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?) [0-1] [0-1] ((\-|\+)?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)((\-|\+)?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?))*)|( ?(Z|z)))*" />
        </xs:restriction>
    </xs:simpleType>

    <!-- Abbreviated Geometry grammar for PatGeometry.Figures -->
    <xs:simpleType name="ST_AbbrGeom">
        <xs:restriction base="xs:string">
            <xs:whiteSpace value="collapse" />
<!--
            <xs:pattern value="( ?(M|m)( ?[rn][scs][rn]))\
                                (\
                                    ( ?(M|m)( ?[rn][scs][rn]))|\
                                    ( ?(L|l)( ?[rn][scs][rn])(
[rn][scs][rn])*)|\
                                    ( ?(H|h|V|v)( ?[rn])( [rn])*)|\
                                    ( ?(Q|q|S|s)( ?[rn][scs][rn]
[rn][scs][rn])(( [rn][scs][rn]){2})*)|\
                                    ( ?(C|c)( ?[rn][scs][rn](
[rn][scs][rn]){2})(( [rn][scs][rn]){3})*)|\
```

```
                                    ( ?(A|a)( ?[rn][scs][rn] [rn] [0-1] [0-1]
[rn][scs][rn])\
                                                    ( [rn][scs][rn] [rn] [0-1]
[0-1] [rn][scs][rn])*)|\
                                    ( ?(Z|z))\
                                )*"/>
-->
            <xs:pattern value="( ?(M|m)( ?((\-|\+)?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)((\-|\+)?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?)))(( ?(M|m)( ?((\-|\+)?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)((\-|\+)?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?)))|( ?(L|l)( ?((\-|\+)?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)((\-|\+)?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?))( ((\-|\+)?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)((\-|\+)?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?))*)|( ?(H|h|V|v)( ?((\-|\+)?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?))( ((\-|\+)?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?))*)|( ?(Q|q|S|s)( ?((\-|\+)?(([0-
9]+(\.[0-9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)((\-|\+)?(([0-
9]+(\.[0-9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?) ((\-|\+)?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)((\-|\+)?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?))(( ((\-|\+)?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)((\-|\+)?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?)){2})*)|( ?(C|c)( ?((\-|\+)?(([0-
9]+(\.[0-9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)((\-|\+)?(([0-
9]+(\.[0-9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?)( ((\-|\+)?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)((\-|\+)?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?)){2})(( ((\-|\+)?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)((\-|\+)?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?)){3})*)|( ?(A|a)( ?((\-|\+)?(([0-
9]+(\.[0-9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)((\-|\+)?(([0-
9]+(\.[0-9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?) ((\-|\+)?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?) [0-1] [0-1] ((\-|\+)?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)((\-|\+)?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?))( ((\-|\+)?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)((\-|\+)?(([0-9]+(\.[0-
9]+)?)|(\.[0-9]+))((e|E)(\-|\+)?[0-9]+)?) ((\-|\+)?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?) [0-1] [0-1] ((\-|\+)?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?)( ?, ?)((\-|\+)?(([0-9]+(\.[0-9]+)?)|(\.[0-
9]+))((e|E)(\-|\+)?[0-9]+)?))*)|( ?(Z|z)))*" />
        </xs:restriction>
    </xs:simpleType>

    <!-- Image reference via Uri -->
    <xs:simpleType name="ST_UriImage">
        <xs:restriction base="xs:anyURI">
            <xs:pattern value="([^\{].*)?" />
        </xs:restriction>
    </xs:simpleType>

    <!-- Image reference via ColorConvertedBitmap -->
    <xs:simpleType name="ST_CtxBmpImage">
        <xs:restriction base="xs:string">
            <xs:pattern
value="\{ColorConvertedBitmap[\s]+[\S]+[\s]+[\S]+\}[\s]*" />
        </xs:restriction>
    </xs:simpleType>

    <!-- Image reference via Uri or ColorConvertedBitmap -->
    <xs:simpleType name="ST_UriCtxBmp">
        <xs:union memberTypes="ST_UriImage ST_CtxBmpImage" />
    </xs:simpleType>
```

```
    <!-- Resource reference -->
    <xs:simpleType name="ST_RscRef">
        <xs:restriction base="xs:string">
            <xs:pattern value="\{StaticResource[\s]+[\S]+\}[\s]*" />
        </xs:restriction>
    </xs:simpleType>

    <!-- Resource reference OR Color -->
    <xs:simpleType name="ST_RscRefColor">
        <xs:union memberTypes="ST_Color ST_RscRef" />
    </xs:simpleType>

    <!-- Resource reference OR Compact Matrix-->
    <xs:simpleType name="ST_RscRefMatrix">
        <xs:union memberTypes="ST_Matrix ST_RscRef" />
    </xs:simpleType>

    <!-- Resource reference OR AbbrGeomF-->
    <xs:simpleType name="ST_RscRefAbbrGeomF">
        <xs:union memberTypes="ST_AbbrGeomF ST_RscRef" />
    </xs:simpleType>

    <!-- Sweep Direction enumeration -->
    <xs:simpleType name="ST_SweepDirection">
        <xs:restriction base="xs:string">
            <xs:enumeration value="Clockwise" />
            <xs:enumeration value="Counterclockwise" />
        </xs:restriction>
    </xs:simpleType>

    <!-- Dash Cap enumeration -->
    <xs:simpleType name="ST_DashCap">
        <xs:restriction base="xs:string">
            <xs:enumeration value="Flat" />
            <xs:enumeration value="Round" />
            <xs:enumeration value="Square" />
            <xs:enumeration value="Triangle" />
        </xs:restriction>
    </xs:simpleType>

    <!-- Line Cap enumeration -->
    <xs:simpleType name="ST_LineCap">
        <xs:restriction base="xs:string">
            <xs:enumeration value="Flat" />
            <xs:enumeration value="Round" />
            <xs:enumeration value="Square" />
            <xs:enumeration value="Triangle" />
        </xs:restriction>
    </xs:simpleType>

    <!-- Line Join enumeration -->
    <xs:simpleType name="ST_LineJoin">
        <xs:restriction base="xs:string">
            <xs:enumeration value="Miter" />
            <xs:enumeration value="Bevel" />
            <xs:enumeration value="Round" />
        </xs:restriction>
    </xs:simpleType>

    <!-- Tile Mode enumeration -->
    <xs:simpleType name="ST_TileMode">
        <xs:restriction base="xs:string">
```

```
                <xs:enumeration value="None" />
                <xs:enumeration value="Tile" />
                <xs:enumeration value="FlipX" />
                <xs:enumeration value="FlipY" />
                <xs:enumeration value="FlipXY" />
            </xs:restriction>
        </xs:simpleType>

        <!-- Color Interpolation Mode enumeration -->
        <xs:simpleType name="ST_ClrIntMode">
            <xs:restriction base="xs:string">
                <xs:enumeration value="ScRgbLinearInterpolation" />
                <xs:enumeration value="SRgbLinearInterpolation" />
            </xs:restriction>
        </xs:simpleType>

        <!-- SpreadMethod Mode enumeration -->
        <xs:simpleType name="ST_SpreadMethod">
            <xs:restriction base="xs:string">
                <xs:enumeration value="Pad" />
                <xs:enumeration value="Reflect" />
                <xs:enumeration value="Repeat" />
            </xs:restriction>
        </xs:simpleType>

        <!-- FillRule Mode enumeration -->
        <xs:simpleType name="ST_FillRule">
            <xs:restriction base="xs:string">
                <xs:enumeration value="EvenOdd" />
                <xs:enumeration value="NonZero" />
            </xs:restriction>
        </xs:simpleType>

        <!-- Edge Mode enumeration -->
        <xs:simpleType name="ST_EdgeMode">
            <xs:restriction base="xs:string">
                <xs:enumeration value="Aliased" />
            </xs:restriction>
        </xs:simpleType>

        <!-- Style Simulation Enumeration -->
        <xs:simpleType name="ST_StyleSimulations">
            <xs:restriction base="xs:string">
                <xs:enumeration value="None" />
                <xs:enumeration value="ItalicSimulation" />
                <xs:enumeration value="BoldSimulation" />
                <xs:enumeration value="BoldItalicSimulation" />
            </xs:restriction>
        </xs:simpleType>

        <!-- ViewUnits Enumeration -->
        <xs:simpleType name="ST_ViewUnits">
            <xs:restriction base="xs:string">
                <xs:enumeration value="Absolute" />
            </xs:restriction>
        </xs:simpleType>

        <!-- MappingMode Enumeration -->
        <xs:simpleType name="ST_MappingMode">
            <xs:restriction base="xs:string">
                <xs:enumeration value="Absolute" />
            </xs:restriction>
```

```
        </xs:simpleType>
</xs:schema>
```

C. Resource Dictionary Key Schema

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema
targetNamespace="http://schemas.microsoft.com/xps/2005/06/resourcedictionary-
key" xmlns="http://schemas.microsoft.com/xps/2005/06/resourcedictionary-key"
xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
blockDefault="#all">

  <xs:attribute name="Key">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <!-- A Key (pattern restriction according to XPS spec) -->
        <xs:pattern
value="(\p{Lu}|\p{Ll}|\p{Lt}|\p{Lo}|\p{Nl}|_)(\p{Lu}|\p{Ll}|\p{Lt}|\p{Lo}|\p{Nl
}|\p{Mn}|\p{Mc}|\p{Nd}|_)*" />
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>

</xs:schema>
```

D. Document Structure Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
targetNamespace="http://schemas.microsoft.com/xps/2005/06/documentstructure"
xmlns="http://schemas.microsoft.com/xps/2005/06/documentstructure"
xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
blockDefault="#all">

      <xs:import namespace="http://www.w3.org/XML/1998/namespace" />

      <!-- ============DocumentStructure Part================ -->
      <!-- Complex Types -->
      <xs:complexType name="CT_DocumentStructure">
            <xs:sequence>
                  <xs:element ref="DocumentStructure.Outline" minOccurs="0"
/>
                  <xs:element ref="Story" minOccurs="0" maxOccurs="unbounded"
/>
            </xs:sequence>
      </xs:complexType>
      <xs:complexType name="CT_CP_Outline">
            <xs:sequence>
                  <xs:element ref="DocumentOutline" />
            </xs:sequence>
      </xs:complexType>
      <xs:complexType name="CT_DocumentOutline">
            <xs:sequence>
                  <xs:element ref="OutlineEntry" maxOccurs="unbounded" />
            </xs:sequence>
            <xs:attributeGroup ref="AG_DocumentOutline" />
      </xs:complexType>
      <xs:complexType name="CT_OutlineEntry">
            <xs:attributeGroup ref="AG_OutlineEntry" />
      </xs:complexType>
      <xs:complexType name="CT_Story">
            <xs:sequence>
                  <xs:element ref="StoryFragmentReference"
maxOccurs="unbounded" />
            </xs:sequence>
            <xs:attributeGroup ref="AG_Story" />
      </xs:complexType>
      <xs:complexType name="CT_StoryFragmentReference">
            <xs:attributeGroup ref="AG_StoryFragmentReference" />
      </xs:complexType>
      <!-- Simple Types -->
      <!-- A Name (ID with pattern restriction according to XPS spec) -->
      <xs:simpleType name="ST_Name">
            <xs:restriction base="xs:string">
                  <xs:pattern
value="(\p{Lu}|\p{Ll}|\p{Lo}|\p{Lt}|\p{Nl})(\p{Lu}|\p{Ll}|\p{Lo}|\p{Lt}|\p{Nl}|
\p{Mn}|\p{Mc}|\p{Nd}|\p{Lm}|_)*" />
            </xs:restriction>
      </xs:simpleType>
      <!-- A Unique Name (ID with pattern restriction according to XPS spec) --
>
      <xs:simpleType name="ST_NameUnique">
            <xs:restriction base="xs:ID">
```

```
                <xs:pattern
value="(\p{Lu}|\p{Ll}|\p{Lo}|\p{Lt}|\p{Nl})(\p{Lu}|\p{Ll}|\p{Lo}|\p{Lt}|\p{Nl}|
\p{Mn}|\p{Mc}|\p{Nd}|\p{Lm}|_)*" />
            </xs:restriction>
    </xs:simpleType>
    <!-- integer greater than or equal to 1 inclusive -->
    <xs:simpleType name="ST_IntGEOne">
            <xs:restriction base="xs:int">
                <xs:minInclusive value="1" />
            </xs:restriction>
    </xs:simpleType>
    <!-- Elements -->
    <xs:element name="DocumentStructure" type="CT_DocumentStructure">
            </xs:element>
    <xs:element name="DocumentStructure.Outline" type="CT_CP_Outline">
            </xs:element>
    <xs:element name="DocumentOutline" type="CT_DocumentOutline">
            </xs:element>
    <xs:element name="OutlineEntry" type="CT_OutlineEntry">
            </xs:element>
    <xs:element name="Story" type="CT_Story">
            </xs:element>
    <xs:element name="StoryFragmentReference"
type="CT_StoryFragmentReference">
            </xs:element>
    <!-- Attribute Groups -->
    <xs:attributeGroup name="AG_DocumentOutline">
            <xs:attribute ref="xml:lang" use="required">
                </xs:attribute>
    </xs:attributeGroup>
    <xs:attributeGroup name="AG_OutlineEntry">
            <xs:attribute name="OutlineLevel" type="ST_IntGEOne"
use="optional" default="1">
                </xs:attribute>
            <xs:attribute name="OutlineTarget" type="xs:anyURI"
use="required">
                </xs:attribute>
            <xs:attribute name="Description" type="xs:string" use="required">
                </xs:attribute>
            <xs:attribute ref="xml:lang" use="optional">
                </xs:attribute>
    </xs:attributeGroup>
    <xs:attributeGroup name="AG_Story">
            <xs:attribute name="StoryName" type="xs:string" use="required">
                </xs:attribute>
    </xs:attributeGroup>
    <xs:attributeGroup name="AG_StoryFragmentReference">
            <xs:attribute name="FragmentName" type="xs:string" use="optional">
                </xs:attribute>
            <xs:attribute name="Page" type="ST_IntGEOne" use="required">
                </xs:attribute>
    </xs:attributeGroup>
    <!-- ================StoryFragments Part============== -->
    <!-- Complex Types -->
    <xs:complexType name="CT_StoryFragments">
            <xs:sequence>
                <xs:element ref="StoryFragment" maxOccurs="unbounded" />
            </xs:sequence>
    </xs:complexType>
    <xs:complexType name="CT_StoryFragment">
            <xs:sequence>
                <xs:element ref="StoryBreak" minOccurs="0" />
```

```
				<xs:choice maxOccurs="unbounded">
					<xs:element ref="SectionStructure" />
					<xs:element ref="ParagraphStructure" />
					<xs:element ref="ListStructure" />
					<xs:element ref="TableStructure" />
					<xs:element ref="FigureStructure" />
				</xs:choice>
				<xs:element ref="StoryBreak" minOccurs="0" />
			</xs:sequence>
			<xs:attributeGroup ref="AG_StoryFragment" />
		</xs:complexType>
		<xs:complexType name="CT_Break">
			</xs:complexType>
		<xs:complexType name="CT_Section">
			<xs:choice maxOccurs="unbounded">
				<xs:element ref="ParagraphStructure" />
				<xs:element ref="ListStructure" />
				<xs:element ref="TableStructure" />
				<xs:element ref="FigureStructure" />
			</xs:choice>
		</xs:complexType>
		<xs:complexType name="CT_Paragraph">
			<xs:choice minOccurs="0" maxOccurs="unbounded">
				<xs:element ref="NamedElement" />
			</xs:choice>
		</xs:complexType>
		<xs:complexType name="CT_Table">
			<xs:choice maxOccurs="unbounded">
				<xs:element ref="TableRowGroupStructure" />
			</xs:choice>
		</xs:complexType>
		<xs:complexType name="CT_TableRowGroup">
			<xs:choice maxOccurs="unbounded">
				<xs:element ref="TableRowStructure" />
			</xs:choice>
		</xs:complexType>
		<xs:complexType name="CT_TableRow">
			<xs:choice maxOccurs="unbounded">
				<xs:element ref="TableCellStructure" />
			</xs:choice>
		</xs:complexType>
		<xs:complexType name="CT_TableCell">
			<xs:choice minOccurs="0" maxOccurs="unbounded">
				<xs:element ref="ParagraphStructure" />
				<xs:element ref="ListStructure" />
				<xs:element ref="TableStructure" />
				<xs:element ref="FigureStructure" />
			</xs:choice>
			<xs:attributeGroup ref="AG_TableCell" />
		</xs:complexType>
		<xs:complexType name="CT_List">
			<xs:choice maxOccurs="unbounded">
				<xs:element ref="ListItemStructure" />
			</xs:choice>
		</xs:complexType>
		<xs:complexType name="CT_ListItem">
			<xs:choice minOccurs="0" maxOccurs="unbounded">
				<xs:element ref="ParagraphStructure" />
				<xs:element ref="ListStructure" />
				<xs:element ref="TableStructure" />
				<xs:element ref="FigureStructure" />
			</xs:choice>
			<xs:attributeGroup ref="AG_ListItem" />
```

```
	</xs:complexType>
	<xs:complexType name="CT_Figure">
		<xs:choice minOccurs="0" maxOccurs="unbounded">
			<xs:element ref="NamedElement" />
		</xs:choice>
	</xs:complexType>
	<xs:complexType name="CT_NamedElement">
		<xs:attributeGroup ref="AG_NamedElement" />
	</xs:complexType>
	<!-- Simple Types -->
	<!-- FragmentType enumeration -->
	<xs:simpleType name="ST_FragmentType">
		<xs:restriction base="xs:string">
			<xs:enumeration value="Content" />
			<xs:enumeration value="Header" />
			<xs:enumeration value="Footer" />
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="ST_Location">
		<xs:restriction base="xs:string">
			<xs:pattern value="([0-9][0-9]*)(\,[0-9][0-9]*)*" />
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="ST_TableSpan">
		<xs:restriction base="xs:int">
			<xs:minInclusive value="1" />
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="ST_ElementIndex">
		<xs:restriction base="xs:int">
			<xs:minInclusive value="0" />
		</xs:restriction>
	</xs:simpleType>
	<!-- Elements -->
	<xs:element name="StoryFragments" type="CT_StoryFragments">
		</xs:element>
	<xs:element name="StoryFragment" type="CT_StoryFragment">
		</xs:element>
	<xs:element name="StoryBreak" type="CT_Break">
		</xs:element>
	<xs:element name="SectionStructure" type="CT_Section">
		</xs:element>
	<xs:element name="ParagraphStructure" type="CT_Paragraph">
		</xs:element>
	<xs:element name="TableStructure" type="CT_Table">
		</xs:element>
	<xs:element name="TableRowGroupStructure" type="CT_TableRowGroup">
		</xs:element>
	<xs:element name="TableRowStructure" type="CT_TableRow">
		</xs:element>
	<xs:element name="TableCellStructure" type="CT_TableCell">
		</xs:element>
	<xs:element name="ListStructure" type="CT_List">
		</xs:element>
	<xs:element name="ListItemStructure" type="CT_ListItem">
		</xs:element>
	<xs:element name="FigureStructure" type="CT_Figure">
		</xs:element>
	<xs:element name="NamedElement" type="CT_NamedElement">
		</xs:element>
	<!-- Attribute Groups -->
	<xs:attributeGroup name="AG_StoryFragment">
```

```
            <xs:attribute name="StoryName" type="xs:string" use="optional">
                  </xs:attribute>
            <xs:attribute name="FragmentName" type="xs:string" use="optional">
                  </xs:attribute>
            <xs:attribute name="FragmentType" type="ST_FragmentType"
use="required">
                  </xs:attribute>
      </xs:attributeGroup>
      <xs:attributeGroup name="AG_TableCell">
            <xs:attribute name="RowSpan" type="ST_TableSpan" use="optional"
default="1">
                  </xs:attribute>
            <xs:attribute name="ColumnSpan" type="ST_TableSpan" use="optional"
default="1">
                  </xs:attribute>
      </xs:attributeGroup>
      <xs:attributeGroup name="AG_ListItem">
            <xs:attribute name="Marker" type="ST_NameUnique" use="optional">
                  </xs:attribute>
      </xs:attributeGroup>
      <xs:attributeGroup name="AG_NamedElement">
            <xs:attribute name="NameReference" type="ST_Name" use="required">
                  </xs:attribute>
      </xs:attributeGroup>
</xs:schema>
```

E. Discard Control Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://schemas.microsoft.com/xps/2005/06/discard-
control" xmlns="http://schemas.microsoft.com/xps/2005/06/discard-control"
xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
blockDefault="#all">

      <xs:complexType name="CT_DiscardControl">
             <xs:sequence>
                    <xs:element ref="Discard" minOccurs="0"
maxOccurs="unbounded" />
             </xs:sequence>
      </xs:complexType>

      <xs:complexType name="CT_Discard">
             <xs:attribute name="SentinelPage" type="xs:anyURI" use="required">
            </xs:attribute>
        <xs:attribute name="Target" type="xs:anyURI" use="required">
            </xs:attribute>
      </xs:complexType>

    <xs:element name="DiscardControl" type="CT_DiscardControl">
        </xs:element>

    <xs:element name="Discard" type="CT_Discard">
        </xs:element>

</xs:schema>
```

F. Abbreviated Geometry Syntax Algorithm

A path geometry specified using the abbreviated geometry syntax (see §11.2.3, "Abbreviated Geometry Syntax," on page 90) is equivalent to a path specified using a path geometry. The following algorithm describes how the abbreviated path syntax can be transformed into a path geometry containing path figures that, in turn, contain various segments.

This algorithm assumes that the presented string is well-formed according to the markup schema. Whitespace skipping is assumed without being explicitly spelled out in the algorithm.

```
Let CURRENTPOINT = 0,0
Create a new PathGeometry PG
PG.FillRule = EvenOdd
Let CURRENTPATHFIGURE = undefined

Read input character CH

if ( CH == 'F' )
{
      Read input character CH
      if ( CH == '0' )
      {
          PG.FillRule = EvenOdd
      }
      else
      {
          PG.FillRule = NonZero
      }
}
else
{
      GOTO label_first
}

label_repeat:
Read input character CH

label_first:

if ( CH == 'm' )
{
      Read relative coordinate pair DX,DY
      Let CURRENTPOINT.X = CURRENTPOINT.X + DX
      Let CURRENTPOINT.Y = CURRENTPOINT.Y + DY
      Create a new PathFigure CURRENTPATHFIGURE and add to PG
      Let attribute CURRENTPATHFIGURE.StartPoint = CURRENTPOINT
}
else if ( CH == 'M' )
{
      Read coordinate pair X,Y
      Let CURRENTPOINT.X = X
      Let CURRENTPOINT.Y = Y
      Create a new PathFigure CURRENTPATHFIGURE and add to PG
      Let attribute CURRENTPATHFIGURE.StartPoint = CURRENTPOINT
}
else if ( CH == 'l' )
      {
            Create new PolyLineSegment S
            Add S to CURRENTPATHFIGURE
label_1:
```

```
                Read relative coordinate pair DX,DY
                Let CURRENTPOINT.X = CURRENTPOINT.X + DX
                Let CURRENTPOINT.Y = CURRENTPOINT.Y + DY
                Add CURRENTPOINT.X,CURRENTPOINT.Y to end of S.Points attribute list
                if ( next character is not a letter )
                {
                    GOTO label_1
                }
           }
           else if ( CH == 'L' )
           {
                Create new PolyLineSegment S
                Add S to CURRENTPATHFIGURE
      label_2:
                Read coordinate pair X,Y
                Let CURRENTPOINT.X = X
                Let CURRENTPOINT.Y = Y
                Add CURRENTPOINT.X,CURRENTPOINT.Y to end of S.Points attribute list
                if ( next character is not a letter )
                {
                      GOTO label_2
                }
           }
           else if ( CH == 'h' )
           {
                Create new PolyLineSegment S
                Add S to CURRENTPATHFIGURE
      label_3:
                Read relative coordinate value DX
                Let CURRENTPOINT.X = CURRENTPOINT.X + DX
                Add CURRENTPOINT.X,CURRENTPOINT.Y to end of S.Points attribute list
                if ( next character is not a letter )
                {
                      GOTO label_3
                }
           }
           else if ( CH == 'H' )
           {
                Create new PolyLineSegment S
                Add S to CURRENTPATHFIGURE
      label_4:
                Read coordinate value X
                Let CURRENTPOINT.X = X
                Add CURRENTPOINT.X,CURRENTPOINT.Y to end of S.Points attribute list
                if ( next character is not a letter )
                {
                      GOTO label_4
                }
           }
           else if ( CH == 'v' )
           {
                Create new PolyLineSegment S
                Add S to CURRENTPATHFIGURE
      label_5:
                Read relative coordinate value DY
                Let CURRENTPOINT.Y = CURRENTPOINT.Y + DY
                Add CURRENTPOINT.X,CURRENTPOINT.Y to end of S.Points attribute list
                if ( next character is not a letter )
                {
                      GOTO label_5
                }
           }
           else if ( CH == 'V' )
           {
                Create new PolyLineSegment S
                Add S to CURRENTPATHFIGURE
      label_6:
                Read coordinate value Y
                Let CURRENTPOINT.Y = Y
                Add CURRENTPOINT.X,CURRENTPOINT.Y to end of S.Points attribute list
```

```
          if ( next character is not a letter )
          {
               GOTO label_6
          }
     }
     else if ( CH == 'c' )
     {
          Create new PolyBezierSegment S
          Add S to CURRENTPATHFIGURE
label_7:
          Read relative coordinate pair DX,DY
          Let POINT.X = CURRENTPOINT.X + DX
          Let POINT.Y = CURRENTPOINT.Y + DY
          Add POINT.X, POINT.Y to end of S.Points attribute list
          Read coordinate pair DX,DY
          Let POINT.X = CURRENTPOINT.X + DX
          Let POINT.Y = CURRENTPOINT.Y + DY
          Add POINT.X, POINT.Y to end of S.Points attribute list
          Read coordinate pair DX,DY
          Let CURRENTPOINT.X = CURRENTPOINT.X + DX
          Let CURRENTPOINT.Y = CURRENTPOINT.Y + DY
          Add CURRENTPOINT.X,CURRENTPOINT.Y to end of S.Points attribute list
          if ( next character is not a letter )
          {
               GOTO label_7
          }
     }
     else if ( CH == 'C' )
     {
          Create new PolyBezierSegment S
          Add S to CURRENTPATHFIGURE
label_8:
          Read coordinate pair X,Y
          Let POINT.X = X
          Let POINT.Y = Y
          Add POINT.X, POINT.Y to end of S.Points attribute list
          Read coordinate pair X,Y
          Let POINT.X = X
          Let POINT.Y = Y
          Add POINT.X, POINT.Y to end of S.Points attribute list
          Read coordinate pair X,Y
          Let CURRENTPOINT.X = X
          Let CURRENTPOINT.Y = Y
          Add CURRENTPOINT.X,CURRENTPOINT.Y to end of S.Points attribute list
          if ( next character is not a letter )
          {
               GOTO label_8
          }
     }
     else if ( CH == 'q' )
     {
          Create new PolyQuadraticBezierSegment S
          Add S to CURRENTPATHFIGURE
label_9:
          Read relative coordinate pair DX,DY
          Let POINT.X = CURRENTPOINT.X + DX
          Let POINT.Y = CURRENTPOINT.Y + DY
          Add POINT.X, POINT.Y to end of S.Points attribute list
          Read relative coordinate pair DX,DY
          Let CURRENTPOINT.X = CURRENTPOINT.X + DX
          Let CURRENTPOINT.Y = CURRENTPOINT.Y + DY
          Add CURRENTPOINT.X,CURRENTPOINT.Y to end of S.Points attribute list
          if ( next character is not a letter )
          {
               GOTO label_9
          }
     }
     else ( if CH == 'Q' )
     {
          Create new PolyQuadraticBezierSegment S
          Add S to CURRENTPATHFIGURE
```

```
label_10:
            Read coordinate pair X,Y
            Let POINT.X = X
            Let POINT.Y = Y
            Add POINT.X, POINT.Y to end of S.Points attribute list
            Read coordinate pair X,Y
            Let CURRENTPOINT.X = X
            Let CURRENTPOINT.Y = Y
            Add CURRENTPOINT.X,CURRENTPOINT.Y to end of S.Points attribute list
            if ( next character is not a letter )
            {
                  GOTO label_10
            }
      }
      else if ( CH == 's' )
      {
            Create new PolyBezierSegment S
            Add S to CURRENTPATHFIGURE
label_11:
            if ( S.Points is non-empty )
            {
                  Let LASTCTRLPOINT = Point before last point in S.Points
                  Let POINT.X = 2 * CURRENTPOINT.X - LASTCTRLPOINT.X
                  Let POINT.Y = 2 * CURRENTPOINT.Y - LASTCTRLPOINT.Y
            }
            else if ( segment before CURRENTPATHSEGMENT is a PolyBezierSegment )
            {
                  Let LASTCTRLPOINT = Point before last point in previous
  PolyBezierSegment
                  Let POINT.X = 2 * CURRENTPOINT.X - LASTCTRLPOINT.X
                  Let POINT.Y = 2 * CURRENTPOINT.Y - LASTCTRLPOINT.Y
            }
            else
            {
                  Let POINT = CURRENTPOINT
            }

            Add POINT.X, POINT.Y to end of S.Points attribute list
            Read relative coordinate pair DX,DY
            Let POINT.X = CURRENTPOINT.X + DX
            Let POINT.Y = CURRENTPOINT.Y + DY
            Add POINT.X, POINT.Y to end of S.Points attribute list
            Read relative coordinate pair DX,DY
            Let CURRENTPOINT.X = CURRENTPOINT.X + DX
            Let CURRENTPOINT.Y = CURRENTPOINT.Y + DY
            Add CURRENTPOINT.X,CURRENTPOINT.Y to end of S.Points attribute list
            if ( next character is not a letter )
            {
                  GOTO label_11
            }
        }
      else if ( CH == 'S' )
      {
            Create new PolyBezierSegment S
            Add S to CURRENTPATHFIGURE
label_12:
            if ( S.Points is non-empty )
            {
                  Let LASTCTRLPOINT = Point before last point in S.Points
                  Let POINT.X = 2 * CURRENTPOINT.X - LASTCTRLPOINT.X
                  Let POINT.Y = 2 * CURRENTPOINT.Y - LASTCTRLPOINT.Y
            }
            else if ( segment before CURRENTPATHSEGMENT is a PolyBezierSegment )
            {
                  Let LASTCTRLPOINT = S.Point before last point in previous
  PolyBezierSegment
                  Let POINT.X = 2 * CURRENTPOINT.X - LASTCTRLPOINT.X
                  Let POINT.Y = 2 * CURRENTPOINT.Y - LASTCTRLPOINT.Y
            }
            else
```

```
                {
                    Let POINT = CURRENTPOINT
                }

                Add POINT.X, POINT.Y to end of S.Points attribute list
                Read coordinate pair X,Y
                Let POINT.X = X
                Let POINT.Y = Y
                Add POINT.X, POINT.Y to end of S.Points attribute list
                Read coordinate pair X,Y
                Let CURRENTPOINT.X = X
                Let CURRENTPOINT.Y = Y
                Add CURRENTPOINT.X,CURRENTPOINT.Y to end of S.Points attribute list
                if ( next character is not a letter )
                {
                    GOTO label_12
                }
          }
          else if ( CH == 'a' )
          {
    label_13:
                Create new ArcSegment S
                Add S to CURRENTPATHFIGURE
                Read Radius Pair RX,RY
                Read Rotation ROT
                Read integer FLAG1
                Read integer FLAG2
                Read relative coordinate pair DX,DY
                Let CURRENTPOINT.X = CURRENTPOINT.X + DX
                Let CURRENTPOINT.Y = CURRENTPOINT.Y + DY
                Let S.Point = CURRENTPOINT.X,CURRENTPOINT.Y
                Let S.IsLargeArc = (FLAG1 == 1 ? true : false)
                Let S.SweepDirection = (FLAG2 == 1 ? Clockwise : Counterclockwise)
                Let S.RotationAngle = ROT
                Let S.Size = RX, RY
                if ( next character is not a letter )
                {
                    GOTO label_13
                }
          }
          else if ( CH == 'A' )
          {
    label_14:
                Create new ArcSegment S
                Add S to CURRENTPATHFIGURE
                Read Radius Pair RX,RY
                Read Rotation ROT
                Read integer FLAG1
                Read integer FLAG2
                Read coordinate pair X,Y
                Let CURRENTPOINT.X = X
                Let CURRENTPOINT.Y = Y
                Let S.Point = CURRENTPOINT.X,CURRENTPOINT.Y
                Let S.IsLargeArc = (FLAG1 == 1 ? true : false)
                Let S.SweepDirection = (FLAG2 == 1 ? Clockwise : Counterclockwise)
                Let S.RotationAngle = ROT
                Let S.Size = RX, RY
                if ( next character is not a letter )
                {
                    GOTO label_14
                }
          }
          else if ( CH == 'z' or CH == 'Z' )
          {
                Let attribute CURRENTPATHFIGURE.IsClosed = true
                Left CURRENTPOINT = First point of first segment of CURRENTPATHFIGURE
                Let CURRENTPATHFIGURE = undefined
          }
    /* This case can not occur, because the input is assumed to be well-formed
      according to the markup schema
          else
```

```
{
     ERROR: Invalid input character
}
*/

if ( End of input reached )
{
     Terminate algorithm, return PG
}
else
{
     GOTO label_repeat
}
```

G. scRGB Gamut Boundary Definition


```xml
<?xml version="1.0" encoding="utf-8" ?>
<cdm:ColorDeviceModel
   ID="http://schemas.microsoft.com/windows/2005/02/color/wcsRGB.cdmp"
   xmlns:cdm="http://schemas.microsoft.com/windows/2005/02/color/
      ColorDeviceModel"
   xmlns:wcs="http://schemas.microsoft.com/windows/2005/02/color/
      WcsCommonProfileTypes"
   xmlns:xs='http://www.w3.org/2001/XMLSchema-instance'>

   <cdm:ProfileName>
      <wcs:Text xml:lang="en-US">wcsRGB virtual device model profile</wcs:Text>
   </cdm:ProfileName>
   <cdm:Description>
      <wcs:Text xml:lang="en-US">wcsRGB virtual s2.13 device model system
profile</wcs:Text>
   </cdm:Description>
   <cdm:Author>
      <wcs:Text xml:lang="en-US">Microsoft Corporation</wcs:Text>
   </cdm:Author>

   <cdm:MeasurementConditions>
      <cdm:ColorSpace>CIEXYZ</cdm:ColorSpace>
      <cdm:WhitePointName>D65</cdm:WhitePointName>
   </cdm:MeasurementConditions>
   <cdm:SelfLuminous>true</cdm:SelfLuminous>
   <cdm:MaxColorant>4.0</cdm:MaxColorant>
   <cdm:MinColorant>-4.0</cdm:MinColorant>

   <cdm:RGBVirtualDevice>
      <cdm:MeasurementData TimeStamp="2005-02-09T22:00:00">
         <cdm:WhitePrimary X="95.05" Y="100.00" Z="108.90" />
         <cdm:RedPrimary X="41.24" Y="21.26" Z="1.93" />
         <cdm:GreenPrimary X="35.76" Y="71.52" Z="11.92" />
         <cdm:BluePrimary X="18.05" Y="7.22" Z="95.05" />
         <cdm:BlackPrimary X="0" Y="0" Z="0" />
         <cdm:Gamma value="1.0" />
         <cdm:GamutBoundarySamples>
            <cdm:RGB R="0.166433" G="-0.016114" B="0.027918" />
            <cdm:RGB R="0.257134" G="-0.026542" B="0.041950" />
            <cdm:RGB R="0.373396" G="-0.039395" B="0.060310" />
            <cdm:RGB R="0.516590" G="-0.054207" B="0.083665" />
            <cdm:RGB R="0.161885" G="-0.013701" B="0.015224" />
            <cdm:RGB R="0.254873" G="-0.024242" B="0.022314" />
            <cdm:RGB R="0.370591" G="-0.036207" B="0.031873" />
            <cdm:RGB R="0.645317" G="-0.053422" B="0.061566" />
            <cdm:RGB R="0.246052" G="-0.020570" B="0.009804" />
            <cdm:RGB R="0.366951" G="-0.033595" B="0.013350" />
            <cdm:RGB R="0.513086" G="-0.047777" B="0.018342" />
            <cdm:RGB R="0.348159" G="-0.027092" B="0.002697" />
            <cdm:RGB R="0.501415" G="-0.043030" B="0.003038" />
            <cdm:RGB R="0.204158" G="-0.007054" B="-0.001921" />
            <cdm:RGB R="0.102843" G="0.005460" B="-0.002574" />
            <cdm:RGB R="0.161680" G="0.005828" B="-0.003751" />
            <cdm:RGB R="0.241218" G="0.005609" B="-0.005263" />
            <cdm:RGB R="0.345298" G="0.004589" B="-0.007162" />
            <cdm:RGB R="1.381166" G="0.758736" B="0.088389" />
            <cdm:RGB R="0.747224" G="0.490147" B="-0.065045" />
```

```
            <cdm:RGB R="0.892023" G="0.585517" B="-0.076908" />
            <cdm:RGB R="1.054700" G="0.693668" B="-0.087994" />
            <cdm:RGB R="1.237403" G="0.815408" B="-0.099715" />
            <cdm:RGB R="0.429786" G="0.463528" B="-0.056073" />
            <cdm:RGB R="0.109329" G="0.215829" B="-0.023359" />
            <cdm:RGB R="0.170974" G="0.342970" B="-0.037798" />
            <cdm:RGB R="0.032781" G="0.182702" B="-0.017604" />
            <cdm:RGB R="0.040703" G="0.236387" B="-0.023182" />
            <cdm:RGB R="0.052010" G="0.299807" B="-0.029289" />
            <cdm:RGB R="-0.015232" G="0.150695" B="-0.012907" />
            <cdm:RGB R="-0.020689" G="0.198705" B="-0.017162" />
            <cdm:RGB R="-0.086126" G="0.133080" B="0.007658" />
            <cdm:RGB R="-0.178409" G="0.235796" B="0.126003" />
            <cdm:RGB R="0.000000" G="0.000000" B="0.000000" />
            <cdm:RGB R="-0.150496" G="0.178902" B="0.156523" />
            <cdm:RGB R="-0.193313" G="0.234293" B="0.204426" />
            <cdm:RGB R="-0.202798" G="0.230449" B="0.290886" />
            <cdm:RGB R="-0.211558" G="0.225780" B="0.384316" />
            <cdm:RGB R="-0.249008" G="0.286162" B="0.470316" />
            <cdm:RGB R="1.024267" G="0.995126" B="0.956518" />
            <cdm:RGB R="-0.211156" G="0.218574" B="0.473918" />
            <cdm:RGB R="-0.209022" G="0.210461" B="0.567755" />
            <cdm:RGB R="0.368209" G="0.814614" B="1.020972" />
            <cdm:RGB R="-0.194449" G="0.199547" B="0.648518" />
            <cdm:RGB R="0.099507" G="0.602350" B="1.001310" />
            <cdm:RGB R="0.447496" G="0.790515" B="1.022822" />
            <cdm:RGB R="-0.123046" G="0.108061" B="0.470343" />
            <cdm:RGB R="-0.150585" G="0.144010" B="0.596962" />
            <cdm:RGB R="0.185831" G="0.575754" B="1.007060" />
            <cdm:RGB R="0.331216" G="0.668936" B="1.023494" />
            <cdm:RGB R="0.410196" G="0.645807" B="1.014063" />
            <cdm:RGB R="0.574938" G="0.752116" B="1.021593" />
            <cdm:RGB R="0.483140" G="0.624458" B="1.005310" />
            <cdm:RGB R="0.635447" G="0.734762" B="1.010061" />
            <cdm:RGB R="-0.052205" G="0.035138" B="0.436365" />
            <cdm:RGB R="1.024267" G="0.995126" B="0.956518" />
            <cdm:RGB R="0.028844" G="-0.005277" B="0.123163" />
            <cdm:RGB R="0.049636" G="-0.010904" B="0.228673" />
            <cdm:RGB R="0.080157" G="-0.012435" B="0.135751" />
            <cdm:RGB R="0.127487" G="-0.020154" B="0.216960" />
            <cdm:RGB R="0.190206" G="-0.029844" B="0.323182" />
            <cdm:RGB R="0.097580" G="-0.013313" B="0.083401" />
            <cdm:RGB R="0.155931" G="-0.021750" B="0.133422" />
            <cdm:RGB R="0.230038" G="-0.031267" B="0.196426" />
            <cdm:RGB R="0.326426" G="-0.043430" B="0.278289" />
            <cdm:RGB R="0.426090" G="-0.050043" B="0.360668" />
            <cdm:RGB R="0.098869" G="-0.010753" B="0.048113" />
            <cdm:RGB R="0.164545" G="-0.019812" B="0.079230" />
            <cdm:RGB R="0.249557" G="-0.030741" B="0.119834" />
            <cdm:RGB R="0.357564" G="-0.043900" B="0.171719" />
            <cdm:RGB R="0.495437" G="-0.060672" B="0.237960" />
            <cdm:RGB R="0.619457" G="-0.065550" B="0.301872" />
            <cdm:RGB R="0.000000" G="0.000000" B="0.000000" />
            <cdm:RGB R="0.169101" G="-0.018572" B="0.047904" />
            <cdm:RGB R="0.255247" G="-0.028467" B="0.071995" />
            <cdm:RGB R="0.368932" G="-0.041653" B="0.103698" />
            <cdm:RGB R="0.508606" G="-0.056762" B="0.143412" />
            <cdm:RGB R="0.637110" G="-0.061275" B="0.186481" />
            <cdm:RGB R="1.249149" G="-0.024436" B="-0.002194" />
            <cdm:RGB R="0.000475" G="-0.001355" B="0.043728" />
            <cdm:RGB R="1.257972" G="-0.026125" B="0.049916" />
            <cdm:RGB R="1.264195" G="-0.027432" B="0.104713" />
```

```
<cdm:RGB R="0.003291" G="-0.009926" B="0.365422" />
<cdm:RGB R="1.287764" G="-0.033804" B="0.414704" />
<cdm:RGB R="0.004536" G="-0.013745" B="0.512430" />
<cdm:RGB R="1.297183" G="-0.037024" B="0.573653" />
<cdm:RGB R="1.307374" G="-0.040929" B="0.763754" />
<cdm:RGB R="0.007818" G="-0.023805" B="0.902837" />
<cdm:RGB R="-0.001325" G="0.002612" B="-0.000199" />
<cdm:RGB R="-0.007337" G="0.015263" B="-0.001058" />
<cdm:RGB R="-0.005430" G="-0.000728" B="0.906022" />
<cdm:RGB R="-0.020327" G="0.043021" B="-0.002887" />
<cdm:RGB R="-0.042091" G="0.089818" B="-0.005932" />
<cdm:RGB R="-0.117963" G="0.253649" B="-0.016500" />
<cdm:RGB R="-0.141327" G="0.262090" B="0.943868" />
<cdm:RGB R="-0.174730" G="0.376455" B="-0.024386" />
<cdm:RGB R="-0.204641" G="0.390245" B="0.954021" />
<cdm:RGB R="-0.245665" G="0.530007" B="-0.034229" />
<cdm:RGB R="-0.282073" G="0.548898" B="0.962351" />
<cdm:RGB R="1.150086" G="0.568927" B="1.001779" />
<cdm:RGB R="-0.374737" G="0.740509" B="0.968358" />
<cdm:RGB R="1.083883" G="0.766726" B="1.001175" />
<cdm:RGB R="1.002593" G="0.985026" B="-0.073045" />
<cdm:RGB R="-0.434478" G="0.938894" B="-0.060390" />
<cdm:RGB R="0.000055" G="-0.000136" B="0.003135" />
<cdm:RGB R="0.000203" G="-0.000555" B="0.016188" />
<cdm:RGB R="0.000896" G="-0.002619" B="0.089419" />
<cdm:RGB R="0.001492" G="-0.004425" B="0.156418" />
<cdm:RGB R="0.002283" G="-0.006839" B="0.247555" />
<cdm:RGB R="0.006038" G="-0.018353" B="0.690852" />
<cdm:RGB R="0.007818" G="-0.023805" B="0.902837" />
<cdm:RGB R="-0.483692" G="0.967403" B="0.971726" />
<cdm:RGB R="0.003728" G="-0.000194" B="0.002807" />
<cdm:RGB R="0.004140" G="-0.000555" B="0.015782" />
<cdm:RGB R="0.004738" G="-0.001328" B="0.043372" />
<cdm:RGB R="0.005489" G="-0.002582" B="0.089155" />
<cdm:RGB R="0.006397" G="-0.004383" B="0.156254" />
<cdm:RGB R="0.007479" G="-0.006795" B="0.247483" />
<cdm:RGB R="0.008758" G="-0.009881" B="0.365433" />
<cdm:RGB R="0.010255" G="-0.013700" B="0.512516" />
<cdm:RGB R="0.011993" G="-0.018308" B="0.691003" />
<cdm:RGB R="0.013995" G="-0.023760" B="0.903048" />
<cdm:RGB R="0.034050" G="-0.018254" B="0.692249" />
<cdm:RGB R="0.036746" G="-0.023689" B="0.904397" />
<cdm:RGB R="0.080545" G="-0.023830" B="0.908226" />
<cdm:RGB R="0.149752" G="-0.024452" B="0.915123" />
<cdm:RGB R="1.002593" G="0.985026" B="-0.073045" />
<cdm:RGB R="1.249149" G="-0.024436" B="-0.002194" />
<cdm:RGB R="1.252780" G="-0.025117" B="0.015342" />
<cdm:RGB R="1.257972" G="-0.026125" B="0.049916" />
<cdm:RGB R="1.271263" G="-0.029093" B="0.182298" />
<cdm:RGB R="1.279122" G="-0.031189" B="0.284927" />
<cdm:RGB R="1.002258" G="0.999362" B="0.998757" />
<cdm:RGB R="0.007818" G="-0.023805" B="0.902837" />
<cdm:RGB R="-0.001325" G="0.002612" B="-0.000199" />
<cdm:RGB R="0.004903" G="-0.018821" B="0.903364" />
<cdm:RGB R="-0.005430" G="-0.000728" B="0.906022" />
<cdm:RGB R="-0.024203" G="0.033431" B="0.912191" />
<cdm:RGB R="-0.042091" G="0.089818" B="-0.005932" />
<cdm:RGB R="-0.052335" G="0.086586" B="0.921578" />
<cdm:RGB R="-0.074175" G="0.159024" B="-0.010406" />
<cdm:RGB R="-0.090966" G="0.161809" B="0.932665" />
<cdm:RGB R="-0.117963" G="0.253649" B="-0.016500" />
<cdm:RGB R="-0.174730" G="0.376455" B="-0.024386" />
<cdm:RGB R="-0.204641" G="0.390245" B="0.954021" />
```

```
<cdm:RGB R="-0.331890" G="0.716723" B="-0.046182" />
<cdm:RGB R="-0.374737" G="0.740509" B="0.968358" />
<cdm:RGB R="-0.434478" G="0.938894" B="-0.060390" />
<cdm:RGB R="0.003567" G="-0.000129" B="-0.000031" />
<cdm:RGB R="0.013995" G="-0.023760" B="0.903048" />
<cdm:RGB R="0.011145" G="-0.018782" B="0.903675" />
<cdm:RGB R="0.001010" G="-0.000696" B="0.906548" />
<cdm:RGB R="-0.017488" G="0.033476" B="0.912853" />
<cdm:RGB R="-0.045318" G="0.086662" B="0.922243" />
<cdm:RGB R="-0.083641" G="0.161924" B="0.933260" />
<cdm:RGB R="-0.133701" G="0.262246" B="0.944373" />
<cdm:RGB R="-0.196722" G="0.390440" B="0.954439" />
<cdm:RGB R="0.020512" G="-0.000542" B="-0.000142" />
<cdm:RGB R="0.034096" G="-0.018719" B="0.905266" />
<cdm:RGB R="0.024587" G="-0.000632" B="0.908671" />
<cdm:RGB R="0.006983" G="0.033602" B="0.915280" />
<cdm:RGB R="-0.019836" G="0.086907" B="0.924618" />
<cdm:RGB R="-0.057114" G="0.162314" B="0.935374" />
<cdm:RGB R="-0.106133" G="0.262787" B="0.946166" />
<cdm:RGB R="-0.168134" G="0.391122" B="0.955918" />
<cdm:RGB R="0.080545" G="-0.023830" B="0.908226" />
<cdm:RGB R="0.078174" G="-0.018848" B="0.909282" />
<cdm:RGB R="0.069557" G="-0.000692" B="0.913080" />
<cdm:RGB R="0.053304" G="0.033705" B="0.919825" />
<cdm:RGB R="0.028089" G="0.087255" B="0.928902" />
<cdm:RGB R="-0.007469" G="0.162946" B="0.939149" />
<cdm:RGB R="-0.054727" G="0.263702" B="0.949360" />
<cdm:RGB R="0.149752" G="-0.024452" B="0.915123" />
<cdm:RGB R="0.147668" G="-0.019423" B="0.916222" />
<cdm:RGB R="0.140009" G="-0.001093" B="0.920043" />
<cdm:RGB R="0.125292" G="0.033611" B="0.926547" />
<cdm:RGB R="0.102016" G="0.087566" B="0.935050" />
<cdm:RGB R="0.068642" G="0.163702" B="0.944526" />
<cdm:RGB R="0.023701" G="0.264902" B="0.953919" />
<cdm:RGB R="0.248626" G="-0.025756" B="0.924685" />
<cdm:RGB R="0.246801" G="-0.020644" B="0.925712" />
<cdm:RGB R="0.240032" G="-0.002034" B="0.929243" />
<cdm:RGB R="0.226828" G="0.033128" B="0.935165" />
<cdm:RGB R="0.205577" G="0.087658" B="0.942824" />
<cdm:RGB R="0.174609" G="0.164417" B="0.951306" />
<cdm:RGB R="0.132331" G="0.266234" B="0.959693" />
<cdm:RGB R="0.381162" G="-0.027864" B="0.936087" />
<cdm:RGB R="0.379554" G="-0.022644" B="0.936982" />
<cdm:RGB R="0.313886" G="0.164903" B="0.959216" />
<cdm:RGB R="0.274412" G="0.267520" B="0.966469" />
<cdm:RGB R="0.551062" G="-0.030854" B="0.948537" />
<cdm:RGB R="0.404402" G="0.399877" B="0.979322" />
<cdm:RGB R="0.761768" G="-0.034778" B="0.961424" />
<cdm:RGB R="1.016507" G="-0.039680" B="0.974324" />
<cdm:RGB R="1.247925" G="-0.024207" B="-0.006804" />
<cdm:RGB R="1.318331" G="-0.045598" B="0.986957" />
<cdm:RGB R="1.002593" G="0.985026" B="-0.073045" />
<cdm:RGB R="-0.017084" G="0.004071" B="0.105192" />
<cdm:RGB R="0.000276" G="-0.000778" B="0.024704" />
<cdm:RGB R="0.001189" G="-0.003497" B="0.124447" />
<cdm:RGB R="0.003835" G="-0.011408" B="0.425034" />
<cdm:RGB R="0.006525" G="-0.019382" B="0.730759" />
<cdm:RGB R="0.991838" G="0.854433" B="0.996305" />
<cdm:RGB R="-0.085296" G="0.980282" B="0.983420" />
<cdm:RGB R="0.997109" G="1.001108" B="1.001729" />
<cdm:RGB R="0.000038" G="-0.000093" B="0.002143" />
<cdm:RGB R="0.000087" G="-0.000230" B="0.006229" />
```

```
<cdm:RGB R="0.000627" G="-0.001818" B="0.062146" />
<cdm:RGB R="0.005522" G="-0.016416" B="0.616931" />
<cdm:RGB R="0.006525" G="-0.019382" B="0.730759" />
<cdm:RGB R="0.001838" G="-0.000115" B="0.001922" />
<cdm:RGB R="0.001968" G="-0.000231" B="0.005969" />
<cdm:RGB R="0.002147" G="-0.000441" B="0.013224" />
<cdm:RGB R="0.002370" G="-0.000761" B="0.024478" />
<cdm:RGB R="0.002634" G="-0.001207" B="0.040491" />
<cdm:RGB R="0.002944" G="-0.001796" B="0.062002" />
<cdm:RGB R="0.003302" G="-0.002545" B="0.089732" />
<cdm:RGB R="0.003715" G="-0.003473" B="0.124383" />
<cdm:RGB R="0.004188" G="-0.004596" B="0.166643" />
<cdm:RGB R="0.004728" G="-0.005932" B="0.217190" />
<cdm:RGB R="0.005341" G="-0.007497" B="0.276684" />
<cdm:RGB R="0.006813" G="-0.011384" B="0.425118" />
<cdm:RGB R="0.008657" G="-0.016392" B="0.617056" />
<cdm:RGB R="0.009736" G="-0.019358" B="0.730905" />
<cdm:RGB R="-0.080359" G="0.980412" B="0.983525" />
<cdm:RGB R="0.005604" G="-0.000209" B="0.001935" />
<cdm:RGB R="0.005786" G="-0.000307" B="0.006004" />
<cdm:RGB R="0.006041" G="-0.000495" B="0.013270" />
<cdm:RGB R="0.006363" G="-0.000795" B="0.024529" />
<cdm:RGB R="0.006745" G="-0.001225" B="0.040547" />
<cdm:RGB R="0.007184" G="-0.001801" B="0.062069" />
<cdm:RGB R="0.007679" G="-0.002542" B="0.089815" />
<cdm:RGB R="0.008231" G="-0.003463" B="0.124487" />
<cdm:RGB R="0.009523" G="-0.005912" B="0.217344" />
<cdm:RGB R="0.010273" G="-0.007474" B="0.276866" />
<cdm:RGB R="0.012011" G="-0.011356" B="0.425354" />
<cdm:RGB R="0.013011" G="-0.013710" B="0.515596" />
<cdm:RGB R="0.014108" G="-0.016361" B="0.617344" />
<cdm:RGB R="0.015308" G="-0.019327" B="0.731220" />
<cdm:RGB R="0.021793" G="-0.013690" B="0.516185" />
<cdm:RGB R="0.023073" G="-0.016337" B="0.617959" />
<cdm:RGB R="0.024454" G="-0.019298" B="0.731861" />
<cdm:RGB R="0.025943" G="-0.022591" B="0.858499" />
<cdm:RGB R="0.037804" G="-0.019298" B="0.732989" />
<cdm:RGB R="0.039522" G="-0.022583" B="0.859653" />
<cdm:RGB R="0.057988" G="-0.022628" B="0.861439" />
<cdm:RGB R="-0.012314" G="0.982157" B="0.984935" />
<cdm:RGB R="0.020932" G="0.982985" B="0.985609" />
<cdm:RGB R="0.110705" G="0.985131" B="0.987376" />
<cdm:RGB R="0.234831" G="0.987909" B="0.989706" />
<cdm:RGB R="0.311005" G="0.989512" B="0.991076" />
<cdm:RGB R="0.494043" G="0.993078" B="0.994197" />
<cdm:RGB R="0.721366" G="0.997001" B="0.997773" />
<cdm:RGB R="0.852925" G="0.999044" B="0.999710" />
<cdm:RGB R="0.997109" G="1.001108" B="1.001729" />
<cdm:RGB R="0.007645" G="-0.022679" B="0.857328" />
<cdm:RGB R="0.001742" G="-0.000069" B="-0.000016" />
<cdm:RGB R="0.010928" G="-0.022656" B="0.857495" />
<cdm:RGB R="0.005453" G="-0.000169" B="-0.000042" />
<cdm:RGB R="0.016619" G="-0.022623" B="0.857831" />
<cdm:RGB R="-0.071870" G="0.980634" B="0.983705" />
<cdm:RGB R="0.012251" G="-0.000330" B="-0.000086" />
<cdm:RGB R="-0.058141" G="0.980991" B="0.983991" />
<cdm:RGB R="-0.038488" G="0.981494" B="0.984397" />
<cdm:RGB R="0.038332" G="-0.000891" B="-0.000240" />
<cdm:RGB R="0.081991" G="-0.022759" B="0.863942" />
<cdm:RGB R="0.112193" G="-0.023008" B="0.867198" />
<cdm:RGB R="0.061770" G="0.983975" B="0.986421" />
<cdm:RGB R="0.168229" G="0.986444" B="0.988470" />
<cdm:RGB R="0.397243" G="0.991242" B="0.992576" />
```

```
                <cdm:RGB R="0.601915" G="0.995006" B="0.995934" />
                <cdm:RGB R="0.721366" G="0.997001" B="0.997773" />
                <cdm:RGB R="0.849328" G="0.852650" B="0.994033" />
                <cdm:RGB R="0.000000" G="0.000000" B="1.000000" />
            </cdm:GamutBoundarySamples>
        </cdm:MeasurementData>

    </cdm:RGBVirtualDevice>

</cdm:ColorDeviceModel>
```

H. Standard Namespaces and Content Types

The following tables list the namespaces and content types used in XPS packages and XPS Documents.

H.1 XML Namespace URIs

Table H–1. Package-wide namespaces

Description	Namespace URI
Content Types	http://schemas.openxmlformats.org/package/2006/content-types
Core Properties	http://schemas.openxmlformats.org/package/2006/metadata/core-properties
Digital Signatures	http://schemas.openxmlformats.org/package/2006/digital-signature
Relationships	http://schemas.openxmlformats.org/package/2006/relationships
Markup Compatibility	http://schemas.openxmlformats.org/markup-compatibility/2006

Table H–2. XPS Document namespaces

Description	Namespace URI
DiscardControl	http://schemas.microsoft.com/xps/2005/06/discard-control
Document Structure	http://schemas.microsoft.com/xps/2005/06/documentstructure
FixedDocument	http://schemas.microsoft.com/xps/2005/06
FixedDocumentSequence	http://schemas.microsoft.com/xps/2005/06
FixedPage	http://schemas.microsoft.com/xps/2005/06
Print Schema Framework	http://schemas.microsoft.com/windows/2003/08/printing/printschemaframework
Print Schema Keywords	http://schemas.microsoft.com/windows/2003/08/printing/printschemakeywords
Resource Dictionary (Key attribute)	http://schemas.microsoft.com/xps/2005/06/resourcedictionary-key
Signature Definitions	http://schemas.microsoft.com/xps/2005/06/signature-definitions
Story Fragments	http://schemas.microsoft.com/xps/2005/06/documentstructure

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

I. Conformance Requirements

This annex is informative

This annex summarizes all conformance requirements for producers and consumers implementing the XML Paper Specification. It is intended as a convenience; the text in the referenced clause or subclause is considered normative in all cases.

Conformance requirements are divided into three tables per clause below, containing the requirements that producers and consumers must follow, those that they should follow, and those that are optional. Each conformance requirement is given a unique ID comprised of a letter (M – MUST; S – SHOULD; O – OPTIONAL), the clause in which the requirement first appears, and a unique ID within that clause. Producers and consumers may use these IDs to report error conditions.

Additionally, the table identifies who is burdened with enforcing or supporting the requirement—the producer of content for that format or the consumer of content in that format—marked with an "×" in the appropriate column. The consumer could be required simply to validate that the producer correctly enforced a requirement in the generation of an XPS Document; these cases are marked with a "v" instead of an "×". In certain cases, a requirement only applies to certain producers or consumers; these are marked with a superscripted letter referenced at the end of the table.

Consumers MUST support the usage of conformance rules marked as "OPTIONAL" and "SHOULD" only for producers if the consumer accesses the referenced feature. If they do not access the referenced feature they must ignore the manifestation of the rule without error.

In addition to the conformance requirements identified below, producers and consumers must meet the conformance requirements described in the OPC.

I.1 XPS Document Format

Table I–1. XPS Document format MUST conformance requirements

ID	Rule	Reference	Producer	Consumer
M1.1	XPS Documents MUST observe all conformance requirements of the OPC specification, except where specifically noted otherwise in this specification.	Clause 1	×	×
M1.2	The XPS Document format MUST use a ZIP archive for its physical model.	1.2	×	×

I.2 Parts and Relationships

Table I–2. Parts and Relationships MUST conformance requirements

ID	Rule	Reference	Producer	Consumer
M2.1	All content to be rendered MUST be contained in the XPS Document.	2.1	×	v
M2.2	Each part contained in an XPS Document MUST use only the appropriate content type.	2.1	×	v[U]
M2.3	An XPS Document MUST contain exactly one FixedDocumentSequence part per fixed payload.	2.1	×	v
M2.4	An XPS Document MUST contain at least one FixedDocument part per fixed payload.	2.1	×	v
M2.5	An XPS Document MUST contain at least one FixedPage part per fixed payload.	2.1	×	v
M2.6	A <Glyphs> element in FixedPage markup MUST reference a Font part that exists in the XPS Document.	2.1	×	v
M2.7	An <ImageBrush> element in FixedPage markup MUST reference an Image part that exists in the XPS Document.	2.1	×	v
M2.8	If FixedPage markup references a Remote Resource Dictionary part, it MUST be included in the XPS Document	2.1	×	v
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.	2.1.1	×	v
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, an editing consumer 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.	2.1.1, 2.1.7.2, 2.1.7.4	×	v[E]
M2.13	Exactly one StartPart relationship is REQUIRED.	2.1.1	×	v
M2.14	The StartPart relationship MUST point from the package to the FixedDocumentSequence part that is the primary fixed payload root.	2.1.2	×	v

M2.15	The order of <DocumentReference> elements in a FixedDocumentSequence part MUST be preserved by editing consumers.	2.1.2		×[E]
M2.16	The order of <PageContent> elements in a FixedDocument MUST be preserved by editing consumers.	2.1.3		×[E]
M2.17	JPEG image parts MUST contain images that conform to the JPEG specification.	2.1.5.1	×	v[U]
M2.18	PNG image parts MUST contain images that conform to the PNG specification.	2.1.5.2	×	v[U]
M2.19	The PNG ancillary chunk tRNS MUST be supported.	2.1.5.2		×
M2.20	The PNG ancillary chunk iCCP MUST be supported.	2.1.5.2		×
M2.21	The PNG ancillary chunk sRGB MUST be ignored.	2.1.5.2		×
M2.22	The PNG ancillary chunk cHRM MUST be ignored.	2.1.5.2		×
M2.23	The PNG ancillary chunk gAMA MUST be ignored.	2.1.5.2		×
M2.24	The PNG ancillary chunk sBIT MUST be ignored.	2.1.5.2		×
M2.25	TIFF image parts MUST contain images that conform to the TIFF specification	2.1.5.3	×	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, "TIFF Images."	2.1.5.3		×
M2.27	If a TIFF file contains multiple image file directories (IFDs), consumers MUST use only the first IFD and ignore all others.	2.1.5.3		×
M2.28	XPS Document consumers MUST support TIFF images using CCITT bilevel encoding.	2.1.5.3		×
M2.29	XPS Document consumers MUST support CMYK TIFF images.	2.1.5.3		×
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.	2.1.5.3		×
M2.31	XPS Document consumers MUST support TIFF images using LZW compression.	2.1.5.3		×
M2.32	XPS Document consumers MUST support TIFF images using differencing predictors.	2.1.5.3		×
M2.33	XPS Document consumers MUST support TIFF images using JPEG compression (compression mode 6 only).	2.1.5.3		×
M2.34	XPS Document consumers MUST support TIFF images with an embedded ICC profile.	2.1.5.3		×
M2.35	Windows Media Photo image files MUST conform to the Windows Media Photo specification.	2.1.5.4	×	v[U]
M2.36	Each FixedPage part MUST NOT have more than one	2.1.6	×	v[T]

	thumbnail part attached.			[T]
M2.37	Thumbnails MUST be either JPEG or PNG images	2.1.6	×	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 and *n*–1 inclusive, where *n* is the number of font faces in the TrueType Collection.	2.1.7	×	v
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.	2.1.7.1	×	v
M2.40	Producers MUST honor the licensing rights specified in OpenType fonts by following the embedding and obfuscation mechanisms described in this specification.	2.1.7.2	×	
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).	2.1.7.2		×
M2.42	For fonts with “Restricted license embedding” licensing intent, producers MUST NOT embed the font.	2.1.7.2	×	
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.	2.1.7.2		×[E]
M2.44	For fonts with “Print and preview embedding” licensing intent, producers MUST perform embedded font obfuscation.	2.1.7.2	×	
M2.45	For fonts with “Print and preview embedding” licensing intent, consumers MUST NOT extract or permanently install the font.	2.1.7.2		×
M2.46	For fonts with “Editable embedding” licensing intent, producers MUST perform embedded font obfuscation.	2.1.7.2	×	
M2.47	For fonts with “Editable embedding” licensing intent, consumers MUST NOT extract or permanently install the font.	2.1.7.2		×
M2.48	For fonts with “No subsetting” licensing intent, producers MUST perform embedded font obfuscation.	2.1.7.2	×	
M2.49	For fonts with “No subsetting” licensing intent, producers MUST NOT subset the font.	2.1.7.2	×	
M2.50	For fonts with “No subsetting” licensing intent, consumers MUST NOT extract or permanently install the font.	2.1.7.2		×
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.	2.1.7.2	×	

M2.52	For fonts with "Bitmap embedding only" licensing intent, consumers MUST NOT extract or permanently install the font.	2.1.7.2		×
M2.53	Producers and consumers MUST perform font obfuscation and de-obfuscation according to the steps described in §9.1.7.3, "Embedded Font Obfuscation."	2.1.7.3	×	×
M2.54	The last segment of the part name for an obfuscated font MUST be the GUID generated during the font obfuscation process, with or without an extension.	2.1.7.3	×	v
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, "Non-Standard Font Compatibility Encoding." All further processing for that font MUST use the selected cmap table.	2.1.7.5	×	×
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.	2.1.7.5		×
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.	2.1.7.5		×
M2.58	Consumers MUST process all PrintTicket parts when an XPS Document is printed.	2.1.9		×[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.	2.1.9.2	×[U]	
M2.60	Consumers MUST process job-level, document-level and page-level settings of PrintTicket parts associated with FixedDocumentSequence parts.	2.1.9.2		×[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.	2.1.9.2		×[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.	2.1.9.2		×[P]

M2.63	When processing a PrintTicket, consumers MUST first remove all levels of PrintTicket content not applicable to the current element.	2.1.9.3		×P
M2.64	When processing a PrintTicket, consumers MUST second validate the PrintTicket according to the methods defined in the PrintTicket Validation Checklist of the Print Schema documentation.	2.1.9.3		×P
M2.65	Following validation of a PrintTicket, the printing consumer MUST properly interpret the print settings according to the rules for merging two PrintTicket parts.	2.1.9.3		×P
M2.66	If there is no print setting merge conflict between different PrintTicket levels, a prefix-scoped element MUST be pushed down, or inherited, from a more general ticket to a more specific ticket. This case is isomorphic to the case where both tickets contain an identical element.	2.1.9.3		×P
M2.67	If there is a print setting merge conflict between different PrintTicket levels, the setting from the most specific ticket MUST take precedence.	2.1.9.3		×P
M2.68	Consumers MUST use semantic document structure provided in included DocumentStructure and StoryFragments parts in preference to any other analysis method of generating such structure.	2.1.11		×
M2.69	Consumers MUST support Markup Compatibility elements and attributes in DocumentStructure, FixedDocument, FixedDocumentSequence, FixedPage, Relationships, Remote Resource Dictionary, SignatureDefinitions, and StoryFragments parts. Before attempting to validate one of these parts against a schema, consumers MUST remove all Markup Compatibility elements and attributes, ignorable namespace declarations, and all ignored elements and attributes not defined in the expected version of XPS Document markup.	2.3.1		×
M2.70	XML content MUST be encoded using either UTF-8 or UTF-16. If any such part includes an encoding declaration (as defined in §4.3.3 of the XML specification), that declaration MUST NOT name any encoding other than UTF-8 or UTF-16.	2.3.2	×	v
M2.71	DTD content MUST NOT be used in the XML markup defined in this specification, and consumers MUST treat the presence of DTD content as an error.	2.3.2	×	×
M2.72	XML content MUST be valid against the corresponding XSD schema defined in this specification. In particular, the XML content MUST NOT contain elements or attributes drawn from namespaces that are not explicitly defined in the corresponding XSD unless the XSD allows elements or attributes drawn from any namespace to be present in particular locations in the XML markup.	2.3.2	×	v

M2.73	XML content MUST NOT contain elements or attributes drawn from “xml” or “xsi” namespaces unless they are explicitly defined in the XSD schema or by other means in the specification.	2.3.2	×	v
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.	2.3.3.2	×	v
M2.75	XPS Document markup MUST NOT use the **xml:space** attribute.	2.3.4	×	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.	2.3.5.1	×	
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.	2.1.1	×[U]	v[U]
M2.78	Consumers MUST support JPEG images that contain the APP1 marker and interpret the EXIF color space correctly.	2.1.5.1		×
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.	2.1.5.3		×

Notes:

E — Only applies to editing consumers.

P — Only applies to printing consumers.

T — Only applies if thumbnails are used by the consumer.

U — Only required for a producer if the part is generated or for a consumer when accessing the part.

Table I–3. Parts and Relationships SHOULD conformance requirements

ID	Rule	Reference	Producer	Consumer
S2.1	It is RECOMMENDED that there be exactly one Required Resource relationship from the FixedPage part for each resource referenced from the markup.	2.1.1	×	

S2.2	Each <DocumentReference> element in a FixedDocumentSequence part SHOULD reference a FixedDocument part by relative URI.	2.1.2	×	
S2.3	Each <PageContent> element in a FixedDocument part SHOULD reference a FixedPage part by relative URI.	2.1.3	×	
S2.4	<ImageBrush> and <Glyphs> elements SHOULD reference Image and Font parts by relative URI.	2.1.4	×	
S2.5	Color profiles embedded in image files SHOULD be used if present and compatible with this specification.	2.1.5		×
S2.6	It is RECOMMENDED that JPEG image part names end with the extension ".jpg".	2.1.5.1	×	
S2.7	The use of CMYK JPEG images is NOT RECOMMENDED. TIFF or Windows Media Photo images SHOULD be used instead to represent CMYK images.	2.1.5.1	×	
S2.8	It is RECOMMENDED that PNG image part names end with the extension ".png".	2.1.5.2	×	
S2.9	It is RECOMMENDED that TIFF image part names end with the extension ".tif".	2.1.5.3	×	
S2.10	Consumers SHOULD ignore unsupported TIFF tags (those not described in Table 9–5 and §9.1.5.3, "TIFF Images"). Producers SHOULD NOT include unsupported tags.	2.1.5.3	×	×
S2.11	Given the wide variety of incompliant TIFF images in 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.	2.1.5.3		×
S2.12	It is RECOMMENDED that Windows Media Photo images end with the extension ".wdp".	2.1.5.4	×	
S2.13	It is RECOMMENDED that if thumbnails are used for pages, a thumbnail SHOULD be included for every page in the document.	2.1.6	×	
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.	2.1.6		×
S2.15	Producers SHOULD use Unicode-encoded fonts.	2.1.7, 2.1.7.5	×	
S2.16	For fonts with "Installable embedding" licensing intent, producers SHOULD perform embedded font obfuscation.	2.1.7.2	×	
S2.17	For fonts with "Installable embedding" licensing intent, consumers SHOULD NOT extract or permanently install the font.	2.1.7.2		×
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	2.1.7.2	×	

	AutomationProperties.Name attribute of the <Path> element.			
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.	2.1.7.3	×	×
S2.20	It is RECOMMENDED that the extension of an obfuscated Font part name be ".odttf" for TrueType fonts and ".odttc" for TrueType collections.	2.1.7.3	×	
S2.21	Producers SHOULD include only PrintTicket settings that support portability of the XPS Document.	2.1.9.1	×	
S2.22	Producers SHOULD only attach PrintTicket parts containing only document-level and page-level settings with FixedDocument parts.	2.1.9.2	×	
S2.23	Producers SHOULD only attach PrintTicket parts containing only page-level settings with FixedPage parts.	2.1.9.3	×	
S2.24	The FixedDocumentSequence part SHOULD follow the part name recommendation "/*<FixedDocSeq>*.fdseq" where *<FixedDocSeq>* is the name of the FixedDocumentSequence. The FixedDocumentSequence SHOULD use the extension ".fdseq".	2.2	×	
S2.25	A FixedDocument part SHOULD follow the part name recommendation "/Documents/*<n>*/*<FixedDocument>*.fdoc" where *<n>* is a numeral that indicates the ordinal position of the fixed document in the fixed document sequence and *<FixedDocument>* is the name of the fixed document. FixedDocument parts SHOULD use the extension ".fdoc".	2.2	×	
S2.26	A FixedPage part SHOULD follow the part name recommendation "/Documents/*<n>*/Pages/*<m>*.fpage" where *<n>* represents the document that includes this page and *<m>* is the page number. FixedPage parts SHOULD use the extension ".fpage".	2.2	×	
S2.27	A resource that is specific to a particular document SHOULD follow the part name recommendation "/Documents/*<n>*/Resources/*<Resource>*" where *<n>* is the document that uses the resource and *<Resource>* is the segments identifying the particular resource. A resource that is shared across multiple documents SHOULD follow the part name recommendation "/Resources/*<Resource>*". A Font resource SHOULD use "Fonts/*<FontName>*.*<FontExt>*" for its *<Resource>* value, where *<FontExt>* SHOULD be either ".ttf" or ".odttf" for non-obfuscated and obfuscated fonts respectively. An Image resource SHOULD use "Images/*<ImageName>*.*<ImageExt>*" for its *<Resource>*	2.2	×	

	value, where <*ImageExt*> is the correct extension for the image type. A Remote Resource Dictionary resource SHOULD use "Dictionaries/<*DictName*>.dict" for its <*Resource*> value. A Remote Resource Dictionary SHOULD use ".dict" as its extension. <*FontName*>, <*ImageName*>, and <*DictName*> SHOULD be a string representation of a GUID value if the resource is a shared resource.		
S2.28	A DocumentStructure part SHOULD follow the part name recommendation "/Documents/<*n*>/Structure/<*DocStruct*>.struct" where <*n*> is the fixed document that the document structure applies to. DocumentStructure parts SHOULD use the extension ".struct".	2.2	×
S2.29	A StoryFragments part SHOULD follow the part name recommendation "/Documents/<*n*>/Structure/Fragments/<*m*>.frag" where <*n*> is the fixed document that contains the story fragments and <*m*> is the page number the StoryFragments part applies to. StoryFragments parts SHOULD use the extension ".frag".	2.2	×
S2.30	ICC profile parts SHOULD follow the part name recommendation "/Documents/<*n*>/Metadata/<*ProfileName*>.<*ProfileExt*>" where <*n*> is the fixed document that contains the profile. ICC profile parts shared across multiple documents SHOULD follow the part name recommendation "/Metadata/<*ProfileName*>.<*ProfileExt*>". In this case, <*ProfileName*> SHOULD be a string representation of a GUID value. The <*ProfileExt*> SHOULD be appropriate to the color profile type, such as ".icm".	2.2	×
S2.31	Thumbnail parts SHOULD follow the part name recommendation "/Documents/<*n*>/Metadata/<*ThumbName*>.<*ThumbExt*>" where <*n*> is the fixed document that contains the profile. If the Thumbnail part represents the package as a whole, it SHOULD follow the part name recommendation "/Metadata/<*ThumbName*>.<*ThumbExt*>". In this case, <*ThumbName*> SHOULD be a string representation of a GUID value. The <*ThumbExt*> SHOULD be appropriate to the image type, either ".png" or ".jpg".	2.2	×
S2.32	PrintTicket part names associated with the entire job SHOULD be associated via relationship with the FixedDocumentSequence part and follow the part name recommendation "/Metadata/<*PrintTicketName*>.xml". PrintTicket parts associated with a particular fixed document	2.2	×

	or fixed page SHOULD follow the part name recommendation "/Documents/<*n*>/Metadata/<*PrintTicketName*>.xml", where <*n*> is the fixed document that uses these parts. PrintTicket parts SHOULD use the extension ".xml".			
S2.33	The names of any non-standard parts that are associated with a particular fixed document SHOULD follow the part name recommendation "/Documents/<*n*>/Other/<*PartName*>", where <*n*> is the fixed document to which the part belongs.	2.2	×	
S2.34	Consumers SHOULD support JPEG images that contain APP0, APP2, APP13, and APP14 markers.	2.1.5.1		×

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	2.1	×	
O2.2	PrintTicket parts MAY be included in an XPS Document.	2.1	×	
O2.3	ICC Profile parts MAY be included in an XPS Document.	2.1	×	
O2.4	DocumentStructure parts MAY be included in an XPS Document.	2.1	×	
O2.5	StoryFragments parts MAY be included in an XPS Document.	2.1	×	
O2.6	SignatureDefinitions parts MAY be included in an XPS Document.	2.1	×	
O2.7	DiscardControl parts MAY be included in an XPS Document.	2.1	×	
O2.8	A Core Properties relationship MAY be included in an XPS Document, from the package to the Core Properties part.	2.1.1	×	
O2.9	A Digital Signatures Origin relationship MAY be included in an XPS Document, from the package to the Digital Signature Origin part.	2.1.1	×	
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.	2.1.1	×	
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.	2.1.1	×	
O2.12	Digital Signature Definitions parts MAY be included in an XPS Document, from a FixedDocument part to the Digital Signature Definitions part.	2.1.1	×	
O2.13	DiscardControl relationships MAY be included in an XPS Document, from the package to a DiscardControl part.	2.1.1	×	
O2.14	DocumentStructure relationships MAY be included in an XPS Document, from a FixedDocument part to the DocumentStructure part.	2.1.1	×	

O2.15	PrintTicket relationships MAY be included in an XPS Document, from a FixedDocumentSequence, FixedDocument, or FixedPage part to a PrintTicket part.	2.1.1	×	
O2.16	StoryFragments relationships MAY be included in an XPS Document, from a FixedPage part to a StoryFragments part.	2.1.1	×	
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.	2.1.1	×	
O2.18	Color Profiles MAY be embedded in image files.	2.1.5	×	
O2.19	Thumbnail images MAY be attached to a FixedPage part using a Thumbnail relationship.	2.1.6	×	
O2.20	Fonts MAY be subsetted based on glyph usage.	2.1.7.1	×	
O2.21	Producers MAY use a 128-bit random number instead of a true GUID for an obfuscated font name.	2.1.7.3	×	
O2.22	An obfuscated Font part MAY have an arbitrary extension.	2.1.7.3	×	
O2.23	Producers MAY add digital signature requests and instructions to an XPS Document in the form of signature definitions.	2.1.10	×	
O2.24	A producer MAY sign against an existing signature definition to provide additional signature information.	2.1.10	×	
O2.25	A recipient of an XPS Document MAY also sign it against a signature definition.	2.1.10	×	
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.	2.1.11		×
O2.28	A resource that is intended to be used across multiple fixed documents MAY be named according to the guidelines for shared resources.	2.2	×	
O2.29	Producers MAY include Markup Compatibility elements and attributes in DocumentStructure, FixedDocument, FixedDocumentSequence, FixedPage, Relationships, Remote Resource Dictionary, SignatureDefinitions, and StoryFragments parts.	2.3.1	×	
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).	2.3.4	×	
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.	2.3.4	×	
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	2.3.4	×	

	behavior specified in the XML Schema Specification.			

I.3 Documents

Table I–5. Document MUST conformance requirements

ID	Rule	Reference	Producer	Consumer
M3.1	The order of <DocumentReference> elements MUST match the order of the documents in the fixed document sequence.	3.1	×	
M3.2	The Source attribute of the <DocumentReference> element MUST specify a FixedDocument part within the XPS Document.	3.1.1	×	v
M3.3	Producers MUST NOT produce a document with multiple <DocumentReference> elements that reference the same fixed document.	3.1.1	×	v
M3.4	The order of <PageContent> elements MUST match the order of the pages in the document.	3.2	×	
M3.5	The Source attribute of the <PageContent> element MUST specify a FixedPage part within the XPS Document.	3.2.1	×	v
M3.6	Producers MUST NOT produce markup where a <PageContent> element references the same fixed page referenced by any other <PageContent> element in the entire XPS Document, even in other fixed documents within the fixed payload.	3.2.1	×	v
M3.7	If the attribute is specified, the **BleedBox** BleedOriginX value MUST be less than or equal to 0.	3.3.1	×	v[F]
M3.8	If the attribute is specified, the **BleedBox** BleedOriginY value MUST be less than or equal to 0.	3.3.1	×	v[F]
M3.9	If the attribute is specified, the **BleedBox** BleedWidth value MUST be greater than or equal to the Width attribute value plus the absolute value of the **BleedBox** BleedOriginX value.	3.3.1	×	v[F]
M3.10	If the attribute is specified, the **BleedBox** BleedHeight value MUST be greater than or equal to the fixed page **Height** value plus the absolute value of the **BleedBox** BleedOriginY value.	3.3.1	×	v[F]
M3.11	If the attribute is specified, the **ContentBox** ContentOriginX value MUST be greater than or equal to 0 and less than the fixed page **Width** attribute value	3.3.2	×	v[F]
M3.12	If the attribute is specified, the **ContentBox** ContentOriginY value MUST be greater than or equal to 0 and less than the fixed page **Height** attribute value.	3.3.2	×	v[F]
M3.13	If the attribute is specified, the **ContentBox** ContentWidth value MUST be less than or equal to the difference between	3.3.2	×	v[F]

	the fixed page **Width** attribute value and the **ContentBox** ContentOriginX value.			
M3.14	If the attribute is specified, the **ContentBox** ContentHeight value MUST be less than or equal to the difference between the fixed page **Height** attribute value and the **ContentBox** ContentOriginY value.	3.3.2	×	v[F]
M3.15	When rendering a fixed page for printing, consumers MUST be aware of the interaction between the fixed page markup and the PrintTicket settings.	3.3.4	×	v[P]
M3.16	When the PrintTicket specifies the page scaling option FitApplicationBleedSizeToPageImageableSize, printing consumers MUST scale the bleed box (producer bleed size) to the PageImageableSize, preserving the aspect ratio.	3.3.4.1		×[P]
M3.17	When the PrintTicket specifies the page scaling option FitApplicationContentSizeToPageImageableSize, printing consumers MUST scale the content box (producer content size) to the PageImageableSize, preserving the aspect ratio.	3.3.4.2		×[P]
M3.18	When the PrintTicket specifies the page scaling option FitApplicationMediaSizeToPageImageableSize, printing consumers MUST scale the height and width (producer media size) to the PageImageableSize, preserving the aspect ratio.	3.3.4.3		×[P]
M3.19	When the PrintTicket specifies the page scaling option FitApplicationMediaSizeToPageMediaSize, printing consumers MUST scale the height and width (producer media size) to the PageMediaSize, preserving the aspect ratio.	3.3.4.4		×[P]
M3.20	The **x:Key** attribute of the <Canvas> element MUST be present when the element is defined in a resource dictionary. It MUST NOT be specified outside a resource dictionary.	3.4	×	v

Notes:

F — Only necessary if the referenced feature is used.

P Only applies to printing consumers.

Table I–6. Document SHOULD conformance requirements

ID	Rule	Reference	Producer	Consumer
S3.1	Specifying a ContentBox attribute for the <FixedPage> element is RECOMMENDED.	3.3	×	
S3.2	Invalid bleed box specifications SHOULD be ignored in favor of the default bleed box.	3.3.1		×
S3.3	Invalid content box specifications SHOULD NOT be rendered and SHOULD generate an error.	3.3.1		×

ID	Rule	Reference	Producer	Consumer
S3.4	In the absence of media scaling, the fixed page content is imaged directly to the physical media with the origin of the fixed page aligned with the origin of the physical media size. Any fixed page content that extends beyond the dimension of the physical media size SHOULD be clipped.	3.3.4		×[P]

Notes:

P — Only applies to printing consumers.

I.4 Graphics

Table I–7. Graphics MUST conformance requirements

ID	Rule	Reference	Producer	Consumer
M4.1	The **x:Key** attribute of the <Path> element MUST be present when the element is defined in a resource dictionary. It MUST NOT be specified outside a resource dictionary.	4.1	×	v
M4.2	The **x:Key** attribute of the <PathGeometry> element MUST be present when the element is defined in a resource dictionary. It MUST NOT be specified outside a resource dictionary.	4.2.1.1	×	v
M4.3	A <PathGeometry> element contains a set of path figures specified either with the **Figures** attribute or with a child <PathFigure> element. Producers MUST NOT specify the path figures of a geometry with both the **Figures** attribute and a child <PathFigure> element.	4.2.1.1	×	v

Table I–8. Graphics SHOULD conformance requirements

ID	Rule	Reference	Producer	Consumer
S4.1	Line segments and curve segments SHOULD NOT be specified as zero-length.	4.2.2.1	×	

Table I–9. Graphics OPTIONAL conformance requirements

ID	Rule	Reference	Producer	Consumer
O4.1	Consumers or viewers that perform anti-aliasing MAY "snap" those control points of the path that are situated on the path bounding box to whole device pixels if the ignorable	4.1		×

	SnapsToDevicePixels attribute is specified as true.

I.5 Text

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.	5.1		×[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.	5.1	×	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.	5.1	×	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.	5.1.3	×	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.	5.1.3	×	
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.	5.1.3	×	
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**.	5.1.4	×	×
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	5.1.4	×	×

	the **Indices** attribute, the consumer MUST display the .notdef glyph			
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 control marks are mapped in the CMAP table.	5.1.4		×
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.	5.1.4		×
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.	5.1.5	×	
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.	5.1.5		×
M5.14	Producers MUST lay out algorithmically italicized glyphs using exactly the same advance widths as when not algorithmically italicized.	5.1.5	×	
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).	5.1.6.2	×	v
M5.16	If a consumer does not understand the specified device font name, it MUST render the embedded version of the font.	5.1.7		×[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.	5.1.7		×[P]
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.	5.1.7		×[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.	5.1.7	×	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.	5.1.7	×	

ID	Rule	Reference	Producer	Consumer
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.	5.1.9		×[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		×

1 **Notes:**

2 F — Only necessary if the referenced feature is used.

3 P — Only applies to printing consumers.

4 S — Only applies to consumers that implement selection.

5 *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.	5.1	×	
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.	5.1.4	×	
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.	5.1.6	×	
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.	5.1.7	×	
S5.5	Specifying a **UnicodeString** for <Glyphs> elements is RECOMMENDED, as it supports searching, selection, and accessibility.	5.1.4	×	

6 *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	5.1.4	×	

	definition within the Unicode specification.			
O5.2	Producers MAY choose to generate **UnicodeString** attribute values that are not normalized by any Unicode-defined algorithm.	5.1.4	×	
O5.3	Consumers that understand the device font name MAY ignore the embedded font and use the device-resident version.	5.1.7		×[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.	5.1.10.1	×	
O5.5	Glyph advance widths MAY be omitted from markup where the advance width desired is specified in the font tables, once adjusted for algorithmic emboldening.	5.1.10.2	×	
O5.6	Glyph horizontal and vertical offsets MAY be omitted from markup where the offset is 0.0.	5.1.10.2	×	

Notes:

P — Only applies to printing consumers.

I.6 Brushes

Table I–13. Brushes MUST conformance requirements

ID	Rule	Reference	Producer	Consumer
M6.1	The **x:Key** attribute of the <SolidColorBrush> element MUST be present when the element is defined in a resource dictionary. It MUST NOT be specified outside a resource dictionary.	6.1	×	v
M6.2	The **x:Key** attribute of the <ImageBrush> element MUST be present when the element is defined in a resource dictionary. It MUST NOT be specified outside a resource dictionary.	6.2	×	v
M6.3	An <ImageBrush> element MUST reference a JPEG, PNG, TIFF 6.0, or Windows Media Photo Image part within the XPS Document package.	6.2	×	v
M6.4	The **x:Key** attribute of the <VisualBrush> element MUST be present when the element is defined in a resource dictionary. It MUST NOT be specified outside a resource dictionary.	6.3	×	v
M6.5	The **x:Key** attribute of the <LinearGradientBrush> element MUST be present when the element is defined in a resource dictionary. It MUST NOT be specified outside a resource dictionary.	6.5	×	v
M6.6	The **x:Key** attribute of the <RadialGradientBrush> element MUST be present when the element is defined in a resource dictionary. It MUST NOT be specified outside a resource dictionary.	6.6	×	v

I.7 Common Properties

Table I–14. Common properties MUST conformance requirements

ID	Rule	Reference	Producer	Consumer
M7.1	Individual resource values MUST be specified within a resource dictionary.	7.2	×	
M7.2	Namespace prefixes in resource definitions MUST apply in the context of the definition, rather than in the context of the resource reference.	7.2.3	×	×
M7.3	An **xml:lang** attribute within a resource definition MUST be interpreted in the context of the resource reference, not the resource definition.	7.2.3	×	×
M7.4	A remote resource dictionary MUST follow the requirements that apply to inline resource dictionaries.	7.2.3.1	×	v
M7.5	A remote resource dictionary MUST NOT contain any resource definition children that reference another remote resource dictionary.	7.2.3.1	×	v
M7.6	A <ResourceDictionary> element that specifies a remote resource dictionary in its **Source** attribute MUST NOT contain any resource definition children.	7.2.3.1	×	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.	7.2.3.1	×	×
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	7.2.5	×	×
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.	7.2.5	×	×
M7.10	If a resource dictionary contains Markup Compatibility elements and attributes, the processing of the Markup Compatibility markup MUST occur in the context of the definition of the resource dictionary, not in the context of resource references.	7.2.6	×	×
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.	7.4.1	×	v

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.	7.2	×	
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.	7.2.3	×	
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.	7.2.3	×	
O7.5	The resource dictionary of a <Canvas> element MAY re-use (and thus override within the scope of the re-use) an **x:Key** value defined in the resource dictionary of a parent or ancestor <Canvas> or <FixedPage> element.	7.2.5	×	
O7.6	A resource definition MAY reference a previously defined resource with the same name that is defined in an ancestor resource dictionary.	7.2.5	×	

I.8 Color

Table I–16. Color MUST conformance requirements

ID	Rule	Reference	Producer	Consumer
M8.1	Consumers MUST support sRGB colors (8 bit-per-channel) in vector data, with and without alpha.	8.1		×
M8.2	Consumers MUST support sRGB colors in image data, using the JPEG, PNG, TIFF, or Windows Media Photo image formats.	8.1		×
M8.3	Consumers MUST support scRGB color specification in vector data, with and without alpha.	8.1		×
M8.4	Consumers MUST support scRGB colors in image data, using the Windows Media Photo image format.	8.1		×
M8.5	Consumers MUST support CMYK colors in vector data.	8.1		×
M8.6	Consumers MUST support CMYK colors in image data, using the TIFF or Windows Media Photo image formats.	8.1		×
M8.7	Consumers MUST support N-Channel colors in vector data.	8.1		×

M8.8	Consumers MUST support N-Channel colors in image data, using the Windows Media Photo image format.	8.1		×
M8.9	Consumers MUST support ICC Version 2 profiles for 3-, 4-, 5-, 6-, 7-, and 8-channel color.	8.1		×
M8.10	Consumers MUST support ICC Version 2 profiles with a Windows Color System (WCS) profile embedded as a private tag, either by handling or ignoring the WCS profile.	8.1		×
M8.11	Consumers MUST inspect the PageDeviceColorSpaceProfileURI PrintTicket setting to determine that this particular color specification is a native device color and MUST NOT be color-managed according to the included profile unless forced to do so for transparency effects.	8.1.7		×[FP]
M8.12	ICC profiles MUST conform to the requirements in the ICC Color Profile specification, Version 3.4.	8.1.8	×[U]	v
M8.13	All ICC profiles used in XPS Documents MUST be either an Input profile, an Output profile, a Monitor (RGB) profile, or a ColorSpace Conversion profile.	8.1.8	×[U]	v
M8.14	Real numbers specified for color channel values of scRGB and ContextColor colors MUST NOT use exponent forms of numbers.	8.1.11	×	v
M8.15	Although alpha values smaller than 0.0 and larger than 1.0 can be specified in scRGB images, the alpha values MUST be clamped to the valid range from 0.0 to 1.0 before any further processing.	8.1.13		×
M8.16	Although alpha values smaller than 0.0 and larger than 1.0 can be specified in CMYK images, the alpha values MUST be clamped to the valid range from 0.0 to 1.0 before any further processing.	8.1.14		×
M8.17	For N-Channel colors, the context color MUST specify a number of channel float values equal to the number of channels in the profile, setting unused ones to 0.0.	8.1.15	×	v
M8.18	Although alpha values smaller than 0.0 and larger than 1.0 can be specified in N-Channel images, the alpha values MUST be clamped to the valid range from 0.0 to 1.0 before any further processing.	8.1.15		×
M8.19	The profile for a named color MUST have 3, 4, 5, 6, 7 or 8 channels (and an *n*CLR signature, where *n* is the number of channels), mapping to a valid PCS.	8.1.16	×[U]	v
M8.20	Although alpha values smaller than 0.0 and larger than 1.0 can be specified in named color images, the alpha values MUST be clamped to the valid range from 0.0 to 1.0 before any further processing.	8.1.16		×
M8.21	Consumers that do not understand a named color MUST compute a color approximation through ICM-compliant color management functions using the specified profile.	8.1.16		×
M8.22	The color name specified by the DocumentImpositionColor	8.3		×[FP]

	PrintTicket setting MUST be matched only to profiles containing exactly one non-zero-length colorant name in the profile’s colorantTable.		
M8.23	The color name specified by the DocumentImpositionColor setting serves as a label for that color only and MUST NOT be matched against any Named Colors known by the consumer.	8.3	×[FP]
M8.24	The comparison of the color name specified by the DocumentImpositionColor PrintTicket setting with the colorant name in the profile’s colorantTable MUST be performed as a case-sensitive ASCII comparison after trimming leading and trailing whitespace from each string.	8.3	×[FP]
M8.25	For gradients, the specified blending color space in the PageBlendColorSpace PrintTicket setting is used only if no gradient stop color values are specified using sRGB or scRGB colors. If any of the gradient stop color values are specified using sRGB or scRGB colors or the consumer does not understand the PageBlendColorSpace PrintTicket settng, the color interpolation mode of the gradient brush MUST be used instead.	8.4	×[FP]
M8.27	If the PageDeviceColorSpaceUsage is set to MatchToDeviceDefault and the profile specified by the PageDeviceColorSpaceURI PrintTicket setting cannot be used as a device color space profile, elements using the profile MUST be color managed like any other element using a color profile.	8.5	×[P]
M8.28	If the PageBlendColorSpace PrintTicket setting is set to ICCProfile, the profile MUST be an output profile, otherwise it MUST be ignored.	8.5	×[FP]
M8.29	Elements using the named color identified by the DocumentImpositionColor PrintTicket setting MUST appear on all color separations.	8.5	×[FP]
M8.30	If an ICC profile is not embedded or associated with a raster image or if the embedded or associated profile is not compatible with the pixel format of the image, the default pixel formats for each color space MUST be treated as defined in §15.2.9, “Color Space Pixel Formats for Raster Images.”	8.2.9	×
M8.31	Channel and tint float values in CMYK, N-Channel, and Named Color syntax MUST be clamped to the valid range from 0.0 to 1.0 before further processing. If the value is used as input for an ICC profile color transformation, it MUST subsequently be linearly scaled to the range from 0 to 255 or from 0 to 65535, depending on whether the profile uses 8-bit or 16-bit input tables.	8.1.14, 8.1.15, 8.1.16	×

Notes:

F — Only necessary if supporting the referenced feature.

P — Only applies to printing consumers.

U — Only required for a producer if the part is generated or for a consumer when accessing the part.

Table I–17. Color SHOULD conformance requirements

ID	Rule	Reference	Producer	Consumer
S8.1	ICC profiles SHOULD be used when embedded in any image format with any color space. Images with integer pixel formats are assumed to have sRGB as the default color space and images with floating point pixel formats are assumed to have scRGB as the default color space; in these cases, an ICC profile is unnecessary.	8.1.8		×
S8.2	If consistency of appearance is important, instead of attaching or embedding a gray ICC profile, the producer SHOULD adjust the gray tone response curve of a grayscale image before adding it to the XPS Document.	8.1.8	×	
S8.3	A producer of XPS documents containing named colors SHOULD create the color profile in such a way that a linear ramp of the channel values corresponding to a named colorant maps to PCS values resulting in the same color appearance for consumers unaware of named colors (or the specific colorant).	8.1.16	×	
S8.4	The ContextColor syntax requires a minimum of 1 Alpha value and 3 Channel values for named colors. It is RECOMMENDED that a 1 or 2 tone profile uses the first 1 or 2 channels, respectively, and specifies 0 for the remaining channels.	8.1.16	×	
S8.5	If the consumer does not know ALL of the colorants named in the clrt tag, it SHOULD treat the profile as if it were a regular N-channel source profile and SHOULD NOT attempt to use any of the known colorants, as that would result in undefined results.	8.1.16		×
S8.6	Support for JPEG CMYK images varies by implementation and SHOULD NOT be used in XPS Documents.	8.2.4.3	×	
S8.7	For consumers that do perform separation, the imposition named color is an indicator that the tint level supplied SHOULD be used for all device colorants.	8.3		×[P]
S8.8	Producers SHOULD create the profile used by the imposition color in such a way that it does not lay down excessive ink when printed on a device that does not perform separation.	8.3	×[P]	
S8.9	If a consumer understands the PageBlendColorSpace PrintTicket setting, it SHOULD convert all color to the specified blending color space before performing a blend operation.	8.4		×[P]

ID	Rule	Reference	Producer	Consumer
S8.10	If the PageDeviceColorSpaceUsage PrintTicket setting is set to MatchToDeviceDefault, the device's internal color profile SHOULD be used for color management of all elements not using the profile specified by the PageDeviceColorSpaceProfileURI PrintTicket setting.	8.5		×[P]
S8.11	If the PageDeviceColorSpaceUsage PrintTicket setting is set to OverrideDeviceDefault and the profile specified by the PageDeviceColorSpaceProfileURI PrintTicet setting has a number of channels matching the number of primaries of the device, it SHOULD be used instead of the device's internal color management for all elements.	8.5		×[P]
S8.12	If the PageBlendColorSpace PrintTicket setting is set to ICCProfile, the Uri property of the option specifies an ICC profile defining the color space that SHOULD be used for blending.	8.5		×[P]
S8.13	The PageICMRenderingIntent PrintTicket setting SHOULD be ignored for elements using a profile that specifies the rendering intent in the profile.	8.5		×[P]
S8.14	A consumer incapable of supporting named colors SHOULD treat the colorant table for named colors tag in an ICC profile as a user-defined custom tag, and therefore ignore it. The consumer SHOULD instead use the color tables as provided in the profile to convert the specified colors to the Profile Connection Space (PCS).	8.1.8		×

Notes:

P — Only applies to printing consumers.

Table I–18. Color OPTIONAL conformance requirements

ID	Rule	Reference	Producer	Consumer
O8.1	Consumers are not required to handle all color spaces natively, but rather MAY convert data specified in a rich color space to sRGB at an early stage (possibly resulting in reduced fidelity).	8.1		×
O8.2	An ICC profile MAY contain the private tag, "MS00", which specifies an embedded Windows Color System (WCS) profile.	8.1.8	×	×
O8.3	When a named color is used in a gradient brush or with transparency, the result produced by consumers that are not named-color aware MAY differ significantly from the result produced by consumers that are named-color aware.	8.1.16		×
O8.4	A named color profile MAY be used with images for spot coloring.	8.1.16	×	×
O8.5	Consumers MAY perform color separation, if desired.	8.3		×
O8.6	Consumers MAY support rich color spaces such as scRGB or CMYK. Consumers that encounter any document using non-sRGB colors MAY process those colors using the simpler sRGB	8.4		×

ID	Rule	Reference	Producer	Consumer
	color space, resulting in deviations, especially for alpha blending			
O8.7	If the PageColorManagement PrintTicket setting specifies a value of Driver, the driver MAY color manage elements or convert them to different color spaces.	8.5		×[P]
O8.8	Elements using the profile specified by PageBlendColorSpace PrintTicket setting with a value of ICCProfile MAY be blended naively (channel-by-channel) without converting through PCS.	8.5		×[P]

Notes:

P — Only applies to printing consumers.

I.9 Document Structure and Interactivity

Table I–19. Document structure MUST conformance requirements

ID	Rule	Reference	Producer	Consumer
M9.1	In order to merge the table cells and rows correctly, producers MUST specify empty <TableCellStructure> elements for cells that do not break across story fragments.	9.1.2	×[D]	
M9.2	If hyperlinked <Path> or <Glyphs> elements are rendered as overlapping on the page, consumers MUST treat the topmost element as the only hyperlink that may be activated in the overlapping region.	9.2.1		×[H]
M9.3	If a producer specifies a **FixedPage.NavigateUri** attribute on a <Canvas> element, consumers MUST treat all child elements of that canvas that do not override this value with their own **FixedPage.NavigateUri** attribute setting as having an associated hyperlink.	9.2.1		×[H]
M9.4	Relative internal hyperlinks between FixedPage parts MUST specify, at a minimum, the named address relative to the FixedDocument part.	9.2.1	×[H]	v[H]
M9.5	In order to be addressable by either a hyperlink or the document outline, the named address MUST appear in the <PageContent.LinkTargets> element in the fixed document.	9.2.1	×[H]	
M9.6	If a named address appears in the <PageContent.LinkTargets> element in the fixed document but is not found in the **Name** attribute of an element within the associated fixed page, consumers MUST treat the top of the associated fixed page as the named address.	9.2.1		×[H]
M9.7	If a named address in a URI fragment is not found, consumers MUST ignore the fragment portion of the URI.	9.2.1		×[H]
M9.8	Internal references MUST specify a page address relative to the fixed document sequence.	9.2.2	×[H]	v[H]

M9.9	Consumers MUST expose every element of the fixed page markup to an accessibility interface in the determined reading order, even if the elements are not referenced in the content structure markup.	9.4.1		×
M9.10	The **Name** attribute MUST NOT be specified on any children of a <ResourceDictionary> element.	9.2.3	×	v

Notes:

D — Only applies to producers that generate document structure or consumers that use it.

H — Only applies to producers that generate hyperlinks or consumers that implement hyperlinks.

Table I–20. Document structure SHOULD conformance requirements

ID	Rule	Reference	Producer	Consumer
S9.1	Every meaningful element in the fixed page markup SHOULD specify a **Name** attribute in order for the document structure markup to refer to it	9.1.1	×	
S9.3	Document structure markup SHOULD NOT refer to a single named element more than once in the document content or to a named element that embeds another named element that it also refers to. When referring to a <Canvas> element, producers SHOULD consider all descendant elements to be referenced in markup order.	9.1.1	×[D]	
S9.4	If a <StoryBreak> element is not present at the beginning of the content structure markup, consumers SHOULD consider the markup a continuation of the previous story fragment that must be merged. Likewise, if a <StoryBreak> element is not present at the end of the content structure markup, consumers SHOULD consider the markup a continuation to the next story fragment that must be merged to determine the cross-fragment content structure.	9.1.2		×[D]
S9.5	Producers authoring document structure information SHOULD reference every element of the fixed page markup that has semantic meaning (such as text or images) in the StoryFragments parts.	9.1.2.2	×[D]	
S9.6	If consumers enable user interactivity, they SHOULD support hyperlink activation and addressing.	9.2		×[H]
S9.7	When activating a hyperlink, consumers SHOULD load the specified resource if they understand the URI type. If the URI is an internal reference to the XPS Document, consumers SHOULD navigate to the URI.	9.2.1		×[H]
S9.8	The value of the **Name** attribute on a <FixedPage>,	9.2.1	×	v

	<Canvas>, <Path>, or <Glyphs> element SHOULD be unique within the scope of the fixed document.			
S9.9	It is RECOMMENDED that **Name** attribute values on <FixedPage>, <Canvas>, <Path>, and <Glyphs> elements be unique within an entire fixed document sequence.	9.2.1	×	v
S9.10	If the **Name** attribute is specified, producers SHOULD also create a corresponding <LinkTarget> element in the FixedDocument part within the <PageContent> element that links to the parent fixed page	9.2.3	×	
S9.11	A hyperlink destination in the same fixed document SHOULD be expressed as a relative URI.	9.2.4	×[H]	
S9.12	If selection is supported, consumers SHOULD provide a visual cue over or around selected elements.	9.3		×[S]
S9.13	Selection order within an XPS Document SHOULD follow reading order.	9.3		×[S]
S9.14	In the absence of document structure provided in the XPS Document, consumers SHOULD, at minimum, rely on the markup order to determine reading order.	9.4.1		×
S9.15	Producers SHOULD order the markup in FixedPage parts to reflect the order in which it is intended to be read.	9.4.1	×	
S9.16	When document structure information is present, consumers SHOULD rely on the order of appearance of named elements in the content structure markup to determine reading order.	9.4.1		×
S9.17	The RECOMMENDED reading order of a page-centric application is 1) order the content by page, 2) order by story fragment within the page based on the order the <StoryFragment> elements are specified in the StoryFragments part for that page, 3) order by <NamedElement> reference within the <StoryFragment> element, 4) append all un-referenced elements that appear in the fixed page markup, ordered by markup order.	9.4.1		×
S9.18	Producers SHOULD order <StoryFragment> elements in each StoryFragments part in their intended reading order.	9.4.1	×[D]	
S9.19	The RECOMMENDED reading order of a story-centric application is as follows: 1) Order content by story in the sequence the <Story> elements appear in the DocumentStructure part. 2) Within a story, order <StoryFragmentReference> elements in the sequence they appear in the DocumentStructure part. 3) Within a story fragment, order by <NamedElement> references in the StoryFragments part markup. 4) Append all un-referenced elements that appear in the fixed page markup, ordered by page number, then markup order	9.4.1		×[D]
S9.20	Producers SHOULD order <Story> elements in the DocumentStructure part in their intended reading order.	9.4.1	×[D]	
S9.21	Producers SHOULD order <StoryFragmentReference>	9.4.1	×[D]	

	elements within a <Story> element in their intended reading order.			
S9.22	A screen reader consumer SHOULD read the document according to its reading order.	9.4.2		×
S9.23	A screen reader SHOULD use the **UnicodeString** attribute of each <Glyphs> element to determine the text to read.	9.4.2		×
S9.24	If a screen reader provides features to navigate the document by structural elements, such as paragraphs or table rows, it SHOULD use any document structure information included in the XPS Document.	9.4.2		×[D]
S9.25	If the screen reader provides features to describe images, it SHOULD read the text provided in the **AutomationProperties.Name** and **AutomationProperties.HelpText** attributes.	9.4.2		×
S9.26	If the screen reader provides features to describe hyperlink addresses, it SHOULD read the text provided in the **FixedPage.NavigateUri** attribute.	9.4.2		×
S9.27	Images and graphics SHOULD specify text alternatives for images and graphics to make this content accessible to vision-impaired individuals. The **AutomationProperties.Name** attribute SHOULD contain a short description of the basic contents of the image or vector graphic. Individual <Path> elements that do not provide any semantic meaning (such as a line between sections or outlining a table) SHOULD NOT specify these text alternative attributes.	9.4.3	×	
S9.28	An image SHOULD specify the **AutomationProperties.Name** and **AutomationProperties.HelpText** attributes on the <Path> element that is filled with an <ImageBrush> that describes the content specified by the ImageSource attribute of the <ImageBrush> element.	9.4.3	×	
S9.29	A vector graphic (a collection of one or more <Path> elements representing a single drawing) SHOULD specify the **AutomationProperties.Name** and **AutomationProperties.HelpText** attributes only once, directly on a <Canvas> element wrapping the <Path> elements comprising the graphic.	9.4.3	×	
S9.30	Children of <VisualBrush> elements SHOULD NOT be referenced by document structure markup.	9.1.2.13	×[D]	

Notes:

D — Only applies to producers that generate document structure or consumers that use it.

H — Only applies to producers that generate hyperlinks or consumers that implement hyperlinks.

S — Only applies to consumers that implement selection.

Table I–21. Document structure OPTIONAL conformance requirements

ID	Rule	Reference	Producer	Consumer
O9.1	Producers MAY choose to add document structure information to XPS Documents. Consumers MAY ignore any authored document structure or hyperlinks.	Clause 9	×[D]	×[D]
O9.2	Producers MAY provide either the document outline or the document content, or both; consumers MAY ignore either or both.	9.1	×[D]	
O9.3	Consumers MAY choose to interpret document structure markup that refers to a single named element more than once, or refers to a named element that embeds another named element that is also referenced, as duplicate content.	9.1.1		×[D]
O9.4	Consumers MAY first attempt to locate named elements for document structure directly from the FixedDocument part markup, where they might appear as <LinkTarget> elements if that named element is also intended as an addressable location.	9.1.1		×[D]
O9.5	A <TableStructure> element is the complete definition of a table. An implementation MAY use it to build special functionality, such as row or column selection.	9.1.2.6		×[D]
O9.6	Internal hyperlinks can specify a named element fragment relative to a particular fixed document, but consumers MAY interpret such a URI relative to the entire fixed document sequence instead	9.2.1		×[H]
O9.7	Consumers MAY ignore the **Name** attribute.	9.2.3		×
O9.8	Consumers MAY ignore the **FixedPage.NavigateUri** attribute.	9.2.4		×
O9.9	Viewing consumers that support interactivity MAY support selection and copying.	9.3		×[S]
O9.10	Consumers MAY use the **FragmentType** attribute of the <StoryFragment> element to determine selection behavior, such as disallowing selection of both the page header and the page contents while allowing independent selection within those stories.	9.3		×[DS]
O9.11	In the absence of document structure information provided in the XPS Document, consumers MAY infer the reading order from the position of elements on the page.	9.4.1		×
O9.12	Consumers MAY use the **FragmentType** attribute of the <StoryFragment> element to determine reading order by interpreting elements that have **FragmentType** values of Header and Footer as belonging first or last in the reading order, respectively.	9.4.1		×[D]

O9.13	Screen readers MAY inspect the **Indices** attribute to resolve potential ambiguities in the **UnicodeString** attribute.	9.4.2		×

Notes:

D — Only applies to producers that generate document structure or consumers that use it.

H — Only applies to producers that generate hyperlinks or consumers that implement hyperlinks.

S — Only applies to consumers that implement selection.

I.10 XPS Document Package Features

Table I–22. XPS Document package feature MUST conformance requirements

ID	Rule	Reference	Producer	Consumer
M10.1	Consumers MUST be prepared to correctly process interleaved packages in which the PrintTicket or the portion of the relationship data attaching the PrintTicket appears in the package after the affected part.	10.1		×[P]
M10.2	Consumers MUST be able to consume packages regardless of their interleaving structure.	10.1.3		×
M10.3	Consumers that lack the resources to process a part MUST indicate an error condition.	10.1.3		×
M10.4	When consuming interleaved packages, consumers MUST NOT discard any parts without instruction from a DiscardControl part unless they have the ability to access the parts again.	10.1.3		×
M10.5	If a consumer encounters a reference to an unknown part, it MUST continue to receive further bytes of the package until the unknown part has been transmitted *or* until the end of the package is reached (indicating an error condition).	10.1.3		×
M10.6	The DiscardControl part MUST NOT reference itself.	10.1.4.1	×	v
M10.7	If either the **Target** attribute or the **SentinelPage** attribute of the <Discard> element contain an invalid reference (refer outside the package), the <Discard> element MUST be ignored.	10.1.4.1.2		×
M10.8	All producers and consumers signing and verifying signatures for end users or applications MUST adhere to the XPS Document signature policy, and producers and consumers MUST interpret digital signatures consistently.	10.2.1	×[A]	×[A]
M10.9	Consumers MUST NOT disable functionality based on	10.2.1		×[A]

	whether or not an action will invalidate a signature.			
M10.10	An XPS Document MUST be considered signed according to the XPS Document signing policy, regardless of the validity of that signature, if the signing rules described in §10.2.1.1, "Signing Rules" are observed.	10.2.1.1		×[A]
M10.11	An XPS Document MUST NOT be considered signed according to the XPS Document signing policy if any part not covered by the signing rules is included in the signature or if any relationship not covered by the signing rules is included in the signature.	10.2.1.1		×[A]
M10.12	An XPS Document digital signer MUST NOT sign an XPS Document that contains content (parts or relationships parts) to be signed that defines the Markup Compatibility namespace but the signer does not fully understand all elements, attributes, and alternate content representations introduced through the markup compatibility mechanisms.	10.2.1.1	×[A]	
M10.13	An XPS Document digital signature MUST be shown as an incompliant digital signature if it violates any of the signing rules regarding parts or relationships that MUST or MUST NOT be signed.	10.2.1.2		×[A]
M10.14	An XPS Document digital signature MUST be shown as a broken digital signature if it is not an incompliant digital signature, but the signature fails the signature validation routines described in the Open Packaging Conventions.	10.2.1.2		×[A]
M10.15	An XPS Document digital signature MUST be shown as a questionable digital signature if it is not an incompliant or broken digital signature, but the certificate cannot be authenticated against the certificate authority or the signed content (parts and relationships) contain elements or attributes from an unknown namespace introduced through Markup Compatibility mechanisms.	10.2.1.2		×[A]
M10.16	An XPS Document digital signature MUST be shown as a valid digital signature if it is not an incompliant, broken, or questionable digital signature.	10.2.1.2		×[A]
M10.17	To prohibit additional signatures in an XPS Document, the signing application MUST sign all the Digital Signature Origin part's relationships of relationship type Digital Signature with the same signature as the rest of the content.	10.2.1.3	×[A]	
M10.18	XPS Document signatures MUST NOT refer to a remote certificate store. All certificates MUST be stored in the XPS Document either as a Certificate part or in the Digital Signature XML Signature part.	10.2.1.4	×[A]	v[A]
M10.19	To link a <SignatureDefinition> to a signature, the value of the **SpotID** MUST be specified in the **Id** attribute of the corresponding <Signature> element in	10.2.2.2.1	×[A]	

ID	Rule	Reference	Producer	Consumer
	the Digital Signature XML Signature part.			
M10.20	Due to space and rendering limitations, producers MUST NOT assume that consumers will use the values specified in the <SpotLocation> element.	10.2.2.3	×[A]	
M10.21	Consumers MUST display the full value of the <Intent> element to the signing party, either in the signature spot or through some other mechanism.	10.2.2.4		×[A]
M10.22	If specified, the <SignBy> date and time MUST be specified as a complete date plus hours, minutes, and seconds in UTC time, as described in the W3C Note "Date and Time Formats."	10.2.2.5	×[A]	
M10.23	There MUST NOT be more than one DiscardControl package relationship.	10.1.4.1	×[R]	v[R]
M10.24	In some cases, producers might rewrite the contents of a package so that parts are provided more than once, allowing consumers to discard a part in order to free resources for additional processing. Each instance of a part MUST be stored as a new, uniquely named part in the package.	10.1.4.1	×[R]	v[R]

Notes:

A — Only applies to producers or consumers implementing digital signature features.

P — Only applies to printing consumers.

R — Only applies to producers or consumers that use DiscardControl parts.

Table I–23. XPS Document package feature SHOULD conformance requirements

ID	Rule	Reference	Producer	Consumer
S10.1	When interleaving, the Content Types stream SHOULD be interleaved according to the recommendations in the OPC specification.	10.1	×	
S10.2	When interleaving, PrintTicket parts SHOULD be written to the package before the part to which they are attached.	10.1	×	
S10.3	When interleaving, the portion of the relationship data attaching the PrintTicket to a part SHOULD be written to the package before the part to which it is attached or in close proximity to the part to which it is attached.	10.1	×	
S10.4	When interleaving, if no PrintTicket settings are specified for a FixedDocumentSequence, FixedDocument, or FixedPage part, an empty PrintTicket part SHOULD be attached to the part, and the portion of the relationship data attaching the empty PrintTicket SHOULD be written to the package before the part to which it is attached or in close proximity to the part to	10.1	×	

	which it is attached.			
S10.5	When interleaving, the last piece of the Relationships part for a FixedPage part SHOULD be written to the package in close proximity to the first piece of the FixedPage part.	10.1	×	
S10.6	It is RECOMMENDED that one empty PrintTicket be shared for all parts that attach an empty PrintTicket.	10.1.1	×	
S10.7	Producers, such as drivers, that target resource-constrained consumers SHOULD: 1) Conservatively model the memory usage of the device. 2) Interleave pieces of parts in the correct order. 3) Decide when certain parts can be discarded by the consumer and inform the consumer within the package stream. 4) Add to the package a uniquely named copy of a resource that could have been discarded, if the resource is referenced by a part sent later in the stream.	10.1.4	×	
S10.8	DiscardControl parts that are not well-formed SHOULD NOT be processed and an error SHOULD NOT be reported.	10.1.4.1		×[U]
S10.9	If a <Discard> element is encountered where either or both of the **Target** attribute and **SentinelPage** attribute identify a part which has not been processed yet (is still unknown), the <Discard> element SHOULD be retained until both parts identified by the **Target** attribute and **SentinelPage** attribute have been processed or until the end of the package is reached.	10.1.4.1.2		×[U]
S10.10	When adding a digital signature to an interleaved package, producers of digitally signed documents that are intended for streaming consumption SHOULD add all digital signature parts and the package relationship to the digital signature parts at the beginning of the package, before adding any other part.	10.1.5	×[A]	
S10.11	Consumers SHOULD inform the end user if an action they are going to take will invalidate an existing signature.	10.2.1		×[A]
S10.12	When printing signed documents, the PrintTicket setting JobDigitalSignatureProcessing SHOULD be used to control the digital signature processing behavior. Consumers SHOULD process this PrintTicket setting, if present	10.2.1.5		×[AP]
S10.13	If the location specified by the <SpotLocation> element is not used when the signature spot is displayed, it is RECOMMENDED that consumers choose a location that does not contain any page content.	10.2.2.3		×[A]
S10.14	It is RECOMMENDED that consumers render signature spots as consistently sized rectangles that include the signer name, the intent, the signing location, and the scope of the XPS Document to be signed.	10.2.2.3		×[A]

S10.15	It is RECOMMENDED that a signature spot be a clickable area used to launch the digital signing process.	10.2.2.3		×[A]
S10.16	If the <SignBy> element is specified, the consumer SHOULD NOT allow the signing party to sign the document using this particular signature spot after the date and time specified.	10.2.2.5	×[A]	
S10.17	The values specified in the Core Properties part SHOULD refer to the entire fixed payload, including the root FixedDocumentSequence part and the compilation of all FixedDocument parts it references.	10.3	×[C]	
S10.18	Head-first XPS Document consumers SHOULD attempt to detect inconsistent packages as soon as possible and SHOULD generate an error message, even if they have already processed the pages that resulted in the error.	10.1		×[K]
S10.19	The viewing consumer SHOULD use the values specified in the <SpotLocation> element to place a signature spot.	10.2.2.3		×[F]

Notes:

A — Only applies to producers or consumers implementing digital signature features.

C — Only applies if Core Properties are used.

F — Only necessary if supporting the referenced feature.

K — Only applies to consumers that consume the XPS Document package head-first.

P — Only applies to printing consumers.

U — Only required for a producer if the part is generated or for a consumer when accessing the part.

Table I–24. XPS Document package feature OPTIONAL conformance requirements

ID	Rule	Reference	Producer	Consumer
O10.1	Interleaving is OPTIONAL.	10.1	×	
O10.2	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.	10.1.2	×	
O10.3	Consumers MAY discard FixedPage parts once they have been processed.	10.1.3		×
O10.4	Consumers MAY discard FixedDocument and FixedDocumentSequence parts after all their child elements and their closing tags have been processed.	10.1.3		×
O10.5	In the absence of explicit directives to the contrary, consumers MAY discard parts as directed by the	10.1.3		×

ID	Rule	Reference	Producer	Consumer
	DiscardControl part.			
O10.6	Some producers (typically drivers) MAY choose a suitable interleaving order by modeling the resource management behavior of the consumer.	10.1.4	×	
O10.7	A consumer MAY decide to ignore a malformed DiscardControl part in its entirety or from the first malformed node onward.	10.1.4.1		×
O10.8	An XPS Document digital signer MAY choose not to sign any content (parts or relationships parts) that defines the Markup Compatibility namespace, even if the content is fully understood.	10.2.1.1	×[A]	
O10.9	An XPS Document digital signature MAY be shown as a questionable digital signature if it is not an incompliant or broken digital signature, but contains some other detectable problem at the discretion of the consumer.	10.2.1.2		×[A]
O10.10	XPS Documents MAY be signed more than once.	10.2.1.3	×[A]	
O10.11	Producers MAY include the JobDigitalSignatureProcessing setting in the job-level PrintTicket within the XPS Document content.	10.2.1.5	×[A]	
O10.12	The SpotID attribute of the <SignatureDefinition> element MAY be used to link to an existing signature.	10.2.2.2.1	×[A]	
O10.13	Consumers MAY choose a size and shape to display a signature spot based on the desired display information and page content.	10.2.2.3		×[A]
O10.14	The <SigningLocation> element MAY be set by the original producer of the XPS Document or by the signing party at the time of signing.	10.2.2.6	×[A]	

Notes:

A — Only applies to producers or consumers implementing digital signature features.

I.11 Rendering Rules

Table I–25. Rendering rules MUST conformance requirements

ID	Rule	Reference	Producer	Consumer
M11.1	Producers MUST generate XPS Documents that can be accurately rendered by following the rules described in the “Rendering Rules” clause. Consumers MUST adhere to the rules described in the “Rendering Rules” clause when rendering XPS Documents	Clause 11	×	×
M11.2	If a non-invertible transform is encountered during rendering, consumers MUST omit rendering the affected	11.1.3		×

	element and all of its child and descendant elements.			
M11.3	If a non-invertible transform is encountered on a geometry (as specified directly on the geometry or through concatenation), the geometry MUST be considered to contain no area.	11.1.3		×
M11.4	Producers MUST NOT assume a specific placement error for curve decomposition or rely on side-effects of a specific consumer implementation.	11.1.5	×	
M11.5	Encountering markup with characteristics outside of the consumer-specific implementation limits MUST cause an error condition.	11.2		×
M11.6	The alpha information in TIFF images using an ExtraSamples tag value of 1 and in Windows Media Photo images using pixel formats WICPixelFormat32bppPBGRA, WICPixelFormat64bppPRGBA or WICPixelFormat128bppPRGBAFloat MUST be interpreted as pre-multiplied alpha information.	11.4.1		×
M11.7	Composition MUST have the same effect as the application of the rules in §18.5, “Composition Rules,” in sequence.	11.5		×
M11.8	The precise source coordinates as specified by the viewbox MUST be used to place an up-sampled image tile, which is equivalent to using fractional pixels of the original source image.	11.7.2		×
M11.9	Consumers MUST precisely position the tiles specified by the image brush and visual brush. If the specified values result in fractional device pixels, the consumer MUST calculate a running placement-error delta and adjust the placement of the next tile where the delta reaches a full device pixel in order to keep the tiles from being increasingly out of phase as the expanse of the path is filled.	11.7.3		×

Table I–26. Rendering rules SHOULD conformance requirements

ID	Rule	Reference	Producer	Consumer
S11.1	All computations on coordinate values SHOULD be performed with at least single floating-point precision. Final conversion (after all transforms have been computed) to device coordinates SHOULD retain at least as much fractional precision as a 28.4 fixed-point representation before performing pixel coverage calculations.	11.1.2	×	×
S11.2	An *ideal* consumer implementation SHOULD render pixels in an 8x8 sub-pixel space, perform an 8x8 box filter sampling, and set the pixel to the resulting color value.	11.1.4		×
S11.3	When rendering a shape, a *practical* implementation (such as a bi-tonal printing device) SHOULD turn on each pixel	11.1.4		×

	whose center (at *x*+0.5) is covered by the shape, or is touched by the shape with the shape extending beyond the pixel center in the positive *x* or *y* direction of the device.		
S11.4	When rendering geometries, consumers SHOULD render curves so they appear smooth from a normal viewing distance.	11.1.5	×
S11.5	When no anti-aliasing is used, abutting shapes that share the same device coordinates for the end-points and control-points of an edge SHOULD be rendered without overlap and without gaps. Ideally, an implementation SHOULD also follow this rule for shapes that are mathematically abutting without sharing device coordinates for end-points and control-points of edges.	11.1.7	×
S11.6	Clipping occurs as if a mask were created from the clip geometry according to the pixel inclusion rules. An ideal consumer SHOULD create such a mask in an 8x8 sub-pixel space and subsequently draw only those sub-pixels of a shape that correspond to "ON" sub-pixels in the mask.	11.1.8	×
S11.7	A practical implementation (such as a bi-tonal printing device) SHOULD create a pixel mask according to the pixel inclusion rules and subsequently draw only those pixels of a shape that correspond to "ON" pixels in the mask. In creating the mask and drawing the shape, the abutment of shapes rule SHOULD be observed so that no pixel of the shape is drawn that would not have been drawn if the clip geometry were another abutting shape.	11.1.8	×
S11.8	A typical consumer SHOULD be able to process markup with the implementation limit characteristics indicated in Table 18–1. Producers SHOULD produce only XPS Documents that stay within these implementation limits.	11.2	×
S11.9	Coordinates are real numbers and SHOULD be computed with at least single floating point precision.	11.2	×
S11.10	If the nesting level of <VisualBrush> elements is higher than 16, a consumer SHOULD attempt to flatten the nested content to a bitmap representation rather than failing to draw.	11.2	×
S11.11	Gradients SHOULD be rendered according to the guidelines described in §18.3, "Gradient Computations."	11.3	×
S11.12	Consumers SHOULD pre-process gradient stops for all gradients using the steps described in §11.3.1.1, "Gradient Stop Pre-Processng."	11.3.1.1	×
S11.13	If any gradient stops use an sRGB or scRGB color specification or the consumer does not understand the PageBlendColorSpace PrintTicket setting, consumers SHOULD blend colors between gradient stops in the color space indicated by the **ColorInterpolationMode** attribute of the gradient brush. If none of the gradient stop	11.3.1.2	×

	elements uses an sRGB or scRGB color specification and the consumer understands the PageBlendColorSpace PrintTicket setting, the PageBlendColorSpace PrintTicket setting SHOULD be used. This SHOULD be a linear, channel-by-channel blend operation.			
S11.14	If a **ColorInterpolationMode** value of SRgbLinearInterpolation is used, the BLEND() function SHOULD convert the color values to sRGB first, and then perform a linear interpolation between them.	11.3.1.2		×
S11.15	If a **ColorInterpolationMode** value of ScRgbLinearInterpolation is used, the BLEND() function SHOULD convert the color values to scRGB first, and then perform a linear interpolation between them.	11.3.1.2		×
S11.16	In the presence of transformations or when individual gradient stops are very close, the local color gradient at the offset used in the BLEND() function may be large, resulting in a large change over the extent of a single device pixel. In this case, it is RECOMMENDED that the BLEND() function interpolate the gradient over the extent of each device pixel. Producers SHOULD NOT, however, rely on a specific effect for such dense gradient specifications.	11.3.1.2	×	×
S11.17	Producers SHOULD either avoid very close gradient stops to the gradient end point when specifying radial gradients where the outside area is visible or avoid specifying radial gradients with a gradient origin on or outside the ellipse (in which case there is no outside area) to ensure consistent rendering results.	11.3.1.2	×	
S11.18	All opacity calculations SHOULD be performed with at least 8-bit precision to provide sufficient quality for nested content.	11.4		×
S11.19	When composing superluminous colors, management of out-of-gamut colors SHOULD be deferred until the result is rendered to the final target, at which point out-of-gamut colors are clipped or color managed.	11.4.1		×
S11.20	The color and appearance of the surface created to hold drawing content as it is composed SHOULD match the destination color and appearance, typically a solid white background for a fixed page or transparent for a canvas.	11.5		×
S11.21	Contours and dashes SHOULD be rendered so that they have the same appearance as if rendered by sweeping the complete length of the contour or dash with a line segment that is perpendicular to the contour and extends with half its length to each side of the contour. All points covered by the sweep of this perpendicular line are part of the dash or contour.	11.6		×
S11.22	Consumers SHOULD ensure that parallel edges of strokes appear parallel.	11.6.1		×
S11.23	Consumers SHOULD produce a visually consistent	11.6.2		×

	appearance of stroke thickness for thin lines, regardless of their orientation or how they fit on the device pixel grid.			
S11.24	Consumers SHOULD select line and curve drawing algorithms that behave symmetrically and result in the same set of device pixels being drawn regardless of the direction of the line or curve (start point and end point exchanged).	11.6.3		×
S11.25	If the current render transform is an invertible matrix, consumers SHOULD perform computations on poly line segments and poly Bézier segments with sufficient accuracy to avoid producing zero-length segments.	11.6.8		×
S11.26	If both width and height of a tile are nearly zero, implementations SHOULD average the color values of the brush contents, resulting in a constant-color brush.	11.7.1		×
S11.27	Producers SHOULD avoid producing extreme cases where either the height, width, or both height and width are nearly zero and SHOULD NOT rely on any specific behavior when they do	11.7.1	×	
S11.28	Source sampling SHOULD be done from the center of the pixel and should be mapped to the center of the pixel in the device-space. With one extent of the viewbox zero, sampling SHOULD be done along a line parallel to the non-zero side. With both extents of the viewbox zero, a point sample SHOULD be taken.	11.7.2		×
S11.29	When up-sampling an image presented at a lower resolution than the device resolution, bilinear filtering SHOULD be used.	11.7.2		×
S11.30	When down-sampling an image presented at a higher resolution than the device resolution, at least a bilinear filter SHOULD be used.	11.7.2		×
S11.31	A stroke using the consistent nominal stroke width convention SHOULD be rendered with a width consistent with other strokes using the convention that have the same **StrokeThickness** attribute value, and consumers aware of this convention SHOULD render such a stroke no thinner than the thinnest visible line that consumer supports without dropouts.			×[F]

F — Only necessary if supporting the referenced feature.

Table I–27. Rendering rules OPTIONAL conformance requirements

ID	Rule	Reference	Producer	Consumer
O11.1	Very high resolution devices MAY use lower fractional precision than a 28.4 fixed-point representation to represent device coordinates.	11.1.2		×

O11.2	Consumers MAY use different rendering logic as long as it closely approximates the logic of rendering pixels in an 8x8 sub-pixel space, performing an 8x8 box filter sampling, and setting the pixel to the resulting color value.	11.1.4		×
O11.3	Devices MAY use sub-pixel masking.	11.1.4		×
O11.4	An implementation capable of anti-aliasing MAY draw a thin line in a way that blends with the background to varying degrees.	11.1.4		×
O11.5	A bi-tonal implementation on a printer MAY draw thin lines with or without drop-outs, or by applying half-toning, depending on the desired output quality.	11.1.4		×
O11.6	Consumers MAY apply pixel placement rules optimized for character rendering to individual glyphs in a <Glyphs> element.	11.1.6		×
O11.7	Behavior of blending with very close gradient stops MAY vary in an implementation-specific manner (see S11.16).	11.3.1.2		×
O11.8	When a radial gradient origin is on or outside the ellipse, the "outside" area (outside the cone defined by the origin and the ellipse) MAY be filled with an interpolated color value, depending on the resolution.	11.3.1.2		×
O11.9	In certain scenarios (such as when rendering 3D scenes to a bitmap), producers MAY choose to create pre-multiplied bitmap data specifying "superluminous" colors.	11.4.1	×	
O11.10	Consumers MAY handle superluminous colors natively or MAY instead choose to convert pre-multiplied source data containing superluminous colors to non-pre-multiplied data before composition by ignoring the superluminous portion of each color channel value.	11.4.1		×
O11.11	A consumer MAY choose always to initialize the alpha channel of the surface created to hold the drawing content as it is composed to 0.0 (transparent) and the color value to black.	11.5		×
O11.12	When doing page composition, if all elements on a canvas and the canvas itself are opaque (an opacity of 1.0) and parent or ancestor <Canvas> elements are also opaque, the elements MAY be drawn directly to the containing fixed page (or canvas), provided all render transform and clip values are observed	11.5.1		×
O11.13	When doing page composition, if an element is fully transparent (an opacity of 0.0), it MAY be skipped.	11.5.1		×
O11.14	When doing page composition, if a canvas has an opacity of 0.0, it and all of its child and descendant elements MAY be skipped.	11.5.1		×
O11.15	When doing page composition, if a canvas has a Clip property with no contained area, the canvas and all of its child and descendant elements MAY be skipped.	11.5.1		×

O11.16	When doing page composition, a consumer MAY further restrict the size of the temporary surface it creates by the effective extent of the geometry specified by the Clip property of the canvas.	11.5.1		×
O11.17	When doing page composition, a consumer MAY use methods to achieve transparency other than creating a temporary surface. Such methods MAY include planar mapping.	11.5.1		×
O11.21	If only one of the width and height values of a tile is nearly zero, the brush should be constant-colored along lines parallel to the narrow side of the viewport, but implementations MAY differ.	11.7.1		×
O11.22	Consumers MAY choose to implement a more sophisticated algorithm for down-sampling an image presented at a higher resolution than the device resolution, such as a Fant scaler, to prevent aliasing artifacts.	11.7.2		×
O11.23	Consumers MAY choose any technique desired to achieve the requirement to precisely place a tile possibly resulting in fractional device pixel placement, such as linear filtering for seams, stretching of the tile (up-sampling or down-sampling), or pre-computing multiple tiles and adjusting behavior according to how the tiles fit on a grid.	11.7.3		×
O11.24	Temporary work canvases MAY be re-used when tiling transparent brushes.	11.7.4		×
O11.25	Producers MAY generate a <Path> element intended to be treated as having a consistent nominal stroke width by specifying the **StrokeDashArray** attribute and by specifying the **StrokeDashOffset** attribute value less than -1.0 times the sum of all the numbers in the **StrokeDashArray** attribute value.	11.6.12	×	

I.12 Additional 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	–	×	v

	by a FixedDocumentSequence part.			
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

BNF of Generic URI Syntax

"BNF of Generic URI Syntax." World Wide Web Consortium. http://www.w3.org/Addressing/URL/5_URI_BNF.html

EXIF Specification

Technical Standardization Committee on AV & IT Storage Systems and Equipment. "Exchangeable Image File Format for Digital Still Cameras: Exif Version 2.2." Japan Electronic Industry Development Association. 2002. http://www.jeita.or.jp

Fant Scaler

Fant, Karl M. "A nonaliasing, real-time spatial transform technique." *IEEE Computer Graphics and Applications* 6 (Jan. 1986): 71-80.

HTML

Jacobs, Ian, Arnaud Le Hors, and Dave Raggett (editors). "HTML 4.01 Specification." World Wide Web Consortium. 1999. http://www.w3.org/TR/1999/REC-html401-19991224/

ICC Color Profile

International Color Consortium. "ICC Profile Format Specification, Version 3.4." 1997. http://www.color.org/icc34.pdf

JPEG

Hamilton, Eric. "JPEG File Interchange Format, Version 1.02." World Wide Web Consortium. 1992. http://www.w3.org/Graphics/JPEG/jfif3.pdf

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

International Telecommunication Union (ITU). "Digital Compression and Coding of Continuous-tone Still Images." 1993. http://www.w3.org/Graphics/JPEG/itu-t81.pdf

Markup Compatibility

Ecma International. "Office Open XML Part 5: Open XML Markup Compatibility." 2006. http://www.ecma-international.org

Open Packaging Conventions

Ecma International. "Office Open XML Part 2: Open Packaging Conventions." 2006. http://www.ecma-international.org

OpenType

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

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

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

PNG

Duce, David (editor). “Portable Network Graphics (PNG) Specification,” Second Edition. World Wide Web Consortium. 2003. http://www.w3.org/TR/2003/REC-PNG-20031110

Print Schema

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

RFC 2119

Bradner, S. “Key words for use in RFCs to Indicate Requirement Levels.” The Internet Society. 1997. http://www.rfc-editor.org

RFC 3066

Alvestrand, H. “Tags for the Identification of Languages.” The Internet Society. 2001. http://www.rfc-editor.org

RFC 4234

Crocker, D. (editor). “Augmented BNF for Syntax Specifications: ABNF.” The Internet Society. 2005. http://www.rfc-editor.org

scRGB

International Electrotechnical Commission (IEC). “Extended RGB colour space—scRGB.” 2003. http://domino.iec.ch/webstore/webstore.nsf/artnum/029678

sRGB

Anderson, Matthew, Srinivasan Chandrasekar, Ricardo Motta, and Michael Stokes. “A Standard Default Color Space for the Internet—sRGB, Version 1.10.” World Wide Web Consortium. 1996. http://www.w3.org/Graphics/Color/sRGB

TIFF

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

Unicode

The Unicode Consortium. The Unicode Standard, Version 4.0.0, defined by: *The Unicode Standard, Version 4.0.* Boston, MA: Addison-Wesley, 2003.

Unicode Character Database

Davis, Mark and Ken Whistler. “Unicode Character Database, Revision 4.0.0.” The Unicode Consortium. 2003. http://www.unicode.org/Public/4.0-Update/UCD-4.0.0.html

WCS Color Profile

Microsoft Corporation. "Windows Color System in Windows Longhorn, WinHEC 2005 Version." 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

XML

Bray, Tim, Eve Maler, Jean Paoli, C. M. Sperberg-McQueen, and François Yergeau (editors). "Extensible Markup Language (XML) 1.0 (Fourth Edition)." World Wide Web Consortium. 2006. http://www.w3.org/TR/2006/REC-xml-20060816/

XML Base

Marsh, Jonathan. "XML Base." World Wide Web Consortium. 2001. http://www.w3.org/TR/2001/REC-xmlbase-20010627/

XML Namespaces

Bray, Tim, Dave Hollander, Andrew Layman, and Richard Tobin (editors). "Namespaces in XML 1.0 (Second Edition)." World Wide Web Consortium. 2006. http://www.w3.org/TR/2006/REC-xml-names-20060816/

XML Schema

Beech, David, Murray Maloney, Noah Mendelsohn, and Henry S. Thompson (editors). "XML Schema Part 1: Structures," Second Edition. World Wide Web Consortium. 2004. http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/

Biron, Paul V. and Ashok Malhotra (editors). "XML Schema Part 2: Datatypes," Second Edition. World Wide Web Consortium. 2004. http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/

K. Index

In the index that follows, italic page numbers are used to indicate illustrations and examples with illustrations. Bold page numbers are used to indicate a primary references when several pages are listed. Page ranges are elided. "*See*" references indicate the primary index location for that topic, while "*See also*" references indicate related index topics.

A

B

C

E

F

G

H

I

J

L

M

N

O

P

R

S

T

V

W

X

Z