[MS-VSDX]:

Visio Graphics Service VSDX File Format

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation ("this documentation") for protocols, file formats, data portability, computer languages, and standards support. Additionally, overview documents cover inter-protocol relationships and interactions.
- **Copyrights**. This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you can make copies of it in order to develop implementations of the technologies that are described in this documentation and can distribute portions of it in your implementations that use these technologies or in your documentation as necessary to properly document the implementation. You can also distribute in your implementation, with or without modification, any schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications documentation.
- **No Trade Secrets**. Microsoft does not claim any trade secret rights in this documentation.
- **Patents**. Microsoft has patents that might cover your implementations of the technologies described in the Open Specifications documentation. Neither this notice nor Microsoft's delivery of this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open Specifications document might be covered by the Microsoft Open Specifications Promise or the Microsoft Community Promise. If you would prefer a written license, or if the technologies described in this documentation are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **License Programs**. To see all of the protocols in scope under a specific license program and the associated patents, visit the Patent Map.
- **Trademarks**. The names of companies and products contained in this documentation might be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.
- **Fictitious Names**. The example companies, organizations, products, domain names, email addresses, logos, people, places, and events that are depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than as specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications documentation does not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments, you are free to take advantage of them. Certain Open Specifications documents are intended for use in conjunction with publicly available standards specifications and network programming art and, as such, assume that the reader either is familiar with the aforementioned material or has immediate access to it.

Support. For questions and support, please contact dochelp@microsoft.com.

1 Introduction

The Visio Graphics Service VSDX File Format describes a Web Drawing, which is a collection of Drawing Pages, Masters, Shapes, Images, Comments, Data Connections, and recalculation information that can be rendered as a drawing.

Sections 1.7 and 2 of this specification are normative. All other sections and examples in this specification are informative.

1.1 Glossary

This document uses the following terms:

add-in: Supplemental functionality that is provided by an external application or macro to extend the capabilities of an application.

American National Standards Institute (ANSI) character set: A **character set** defined by a code page approved by the American National Standards Institute (ANSI). The term "ANSI" as used to signify Windows code pages is a historical reference and a misnomer that persists in the Windows community. The source of this misnomer stems from the fact that the Windows code page 1252 was originally based on an ANSI draft, which became International Organization for Standardization (ISO) Standard 8859-1 [ISO/IEC-8859-1]. In Windows, the ANSI character set can be any of the following code pages: 1252, 1250, 1251, 1253, 1254, 1255, 1256, 1257, 1258, 874, 932, 936, 949, or 950. For example, "ANSI application" is usually a reference to a non-**Unicode** or code-page-based application. Therefore, "ANSI character set" is often misused to refer to one of the character sets defined by a Windows code page that can be used as an active system code page; for example, character sets defined by code page 1252 or character sets defined by code page 950. Windows is now based on Unicode, so the use of ANSI character sets is strongly discouraged unless they are used to interoperate with legacy applications or legacy data.

assembly name: The name of a collection of one or more files that is versioned and deployed as a unit. See also assembly.

Augmented Backus-Naur Form (ABNF): A modified version of Backus-Naur Form (BNF), commonly used by Internet specifications. ABNF notation balances compactness and simplicity with reasonable representational power. ABNF differs from standard BNF in its definitions and uses of naming rules, repetition, alternatives, order-independence, and value ranges. For more information, see [RFC5234].

bitmap (BMP): A representation of characters or graphics by individual pixels. The pixels can be arranged in rows (horizontal) and columns (vertical). Each pixel can be represented by one or more bits.

Boolean: An operation or expression that can be evaluated only as either true or false.

character set: A mapping between the characters of a written language and the values that are used to represent those characters to a computer.

class name: The name that is used to refer to a class module that provides an implementation of a behavior.

color space: A system that describes color numerically by mapping color components to a multidimensional coordinate system. The number of dimensions is typically two, three, or four. For example, if colors are expressed as a combination of the three components red, green, and blue, a three-dimensional space can describe all possible colors. Grayscale colors can be mapped to a two-dimensional color space. If transparency is considered a component, four dimensions are appropriate. Also referred to as color model.

connection string: A series of arguments, delimited by a semicolon, that defines the location of a database and how to connect to it.

culture name: A part of a language identification tagging system, as described in [RFC1766]. Culture names adhere to the format "<languagecode2>-<country/regioncode2>." If a two-letter language code is not available, a three-letter code that is derived from [ISO-639] is used.

data provider: A known data source that is specific to a target type and that provides data to a collector type.

data source: A database, web service, disk, file, or other collection of information from which data is queried or submitted. Supported data sources vary based on application and data provider.

drawing: A collection of drawing objects, such as shapes, curves, or WordArt, that are viewed together as a single image.

embedded image: An image that is stored within a document rather than being linked to a source file that is outside the document.

embedded object: An object that is created by using one application and is hosted in a document that was created by using another application. Embedding an object, rather than inserting or pasting it, ensures that the object retains its original format. Users can double-click an embedded object and edit it with the toolbars and menus from the application that was used to create it. See also Object Linking and Embedding (OLE).

enhanced metafile format (EMF): A file format that supports the device-independent definitions of images.

field: An element or attribute in a data source that can contain data.

floating-point number: A number that is represented by a mantissa and an exponent according to a given base. The mantissa is typically a value between "0" and "1". To find the value of a floating-point number, the base is raised to the power of the exponent, and the mantissa is multiplied by the result.

font: An object that defines the graphic design, or formatting, of a collection of numbers, symbols, and letters. A font specifies the style (such as bold and strikeout), size, family (a typeface such as Times New Roman), and other qualities to describe how the collection is drawn.

gamma correction: In digital imaging, the process of changing the brightness, contrast, or color balance of an image by assigning new values (different colors) to gray or color tones.

globally unique identifier (GUID): A term used interchangeably with universally unique identifier (UUID) in Microsoft protocol technical documents (TDs). Interchanging the usage of these terms does not imply or require a specific algorithm or mechanism to generate the value. Specifically, the use of this term does not imply or require that the algorithms described in [RFC4122] or [C706] have to be used for generating the GUID. See also universally unique identifier (UUID).

Graphics Interchange Format (GIF): A compression format that supports device-independent transmission and interchange of bitmapped image data. The format uses a palette of up to 256 distinct colors from the 24-bit **RGB** color space. It also supports animation and a separate palette of 256 colors for each frame. The color limitation makes the GIF format unsuitable for reproducing color photographs and other images with gradients of color, but it is well-suited for simpler images such as graphics with solid areas of color.

header row: A row in a table, typically the first row, that contains labels for columns in the table.

hue-saturation-luminance (HSL): A color model that defines a color by using three dimensions: hue, the color itself; saturation, the purity of the color; and luminance, the amount of light that is either reflected or absorbed by the color. See also color scheme and **color space**.

hyperlink location: A portion of a hyperlink that specifies the location of a specific item, such as a bookmark, within a document, object, or other type of resource; for example "#bookmark" in the hyperlink location C:\Documents\Document.docx#bookmark.

Hypertext Transfer Protocol (HTTP): An application-level protocol for distributed, collaborative, hypermedia information systems (text, graphic images, sound, video, and other multimedia files) on the World Wide Web.

Joint Photographic Experts Group (JPEG): A raster graphics file format for displaying high-resolution color graphics. JPEG graphics apply a user-specified compression scheme that can significantly reduce the file sizes of photo-realistic color graphics. A higher level of compression results in lower quality, whereas a lower level of compression results in higher quality. JPEG-format files have a .jpg or .jpeg file name extension.

language code identifier (LCID): A 32-bit number that identifies the user interface human language dialect or variation that is supported by an application or a client computer.

list: An organization of a region of cells into a tabular structure in a workbook.

metafile: A file that stores an image as graphical objects, such as lines, circles, and polygons, instead of pixels. A metafile preserves an image more accurately than pixels when an image is resized.

Office data connection (ODC) file: A file that stores information about a connection to a data source, such as an Access database, worksheet, or text file. This file facilitates data source administration.

OLE DB: A set of interfaces that are based on the Component Object Model (COM) programming model and expose data from a variety of sources. These interfaces support the amount of Database Management System (DBMS) functionality that is appropriate for a data store and they enable a data store to share data.

Open Database Connectivity (ODBC): A standard software API method for accessing data that is stored in a variety of proprietary personal computer, minicomputer, and mainframe databases. It is an implementation of [ISO/IEC9075-3:2008] and provides extensions to that standard.

Portable Network Graphics (PNG): A bitmap graphics file format that uses lossless data compression and supports variable transparency of images (alpha channels) and control of image brightness on different computers (gamma correction). PNG-format files have a .png file name extension.

primary key: A field or set of fields that uniquely identifies each record in a table. A primary key cannot contain a null value.

query: A formalized instruction to a data source to either extract data or perform a specified action. A query can be in the form of a query expression, a method-based query, or a combination of the two. The data source can be in different forms, such as a relational database, XML document, or in-memory object. See also search query.

red-green-blue (RGB): A color model that describes color information in terms of the red (R), green (G), and blue (B) intensities in a color.

row: A single set of data that is displayed horizontally in a worksheet or a table.

Tagged Image File Format (TIFF): A high-resolution, tag-based graphics format. TIFF is used for the universal interchange of digital graphics.

text run: A string of characters that represents a discrete span of text with the same formatting properties.

token: A word in an item or a search query that translates into a meaningful word or number in written text. A token is the smallest textual unit that can be matched in a search query. Examples include "cat", "AB14", or "42".

Unicode: A character encoding standard developed by the Unicode Consortium that represents almost all of the written languages of the world. The **Unicode** standard [UNICODE5.0.0/2007] provides three forms (UTF-8, UTF-16, and UTF-32) and seven schemes (UTF-8, UTF-16, UTF-16 BE, UTF-16 LE, UTF-32, UTF-32 LE, and UTF-32 BE).

Uniform Resource Identifier (URI): A string that identifies a resource. The URI is an addressing mechanism defined in Internet Engineering Task Force (IETF) Uniform Resource Identifier (URI): Generic Syntax [RFC3986].

Uniform Resource Locator (URL): A string of characters in a standardized format that identifies a document or resource on the World Wide Web. The format is as specified in [RFC1738].

UTF-16: A standard for encoding Unicode characters, defined in the Unicode standard, in which the most commonly used characters are defined as double-byte characters. Unless specified otherwise, this term refers to the UTF-16 encoding form specified in [UNICODE5.0.0/2007] section 3.9.

view: See form view (Microsoft InfoPath), list view (SharePoint Products and Technologies), or **View** (Microsoft Business Connectivity Services).

whitespace: A character that can be found between words, including a space (" "), a carriage return in combination with a line feed (newline), and a tab character.

workbook: A container for a collection of sheets.

zero-based index: An index in which the first item has an index of "0" (zero).

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as defined in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the Errata.

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information.

[GIF89a] CompuServe Incorporated, "Graphics Interchange Format(sm)", Graphics Interchange Format Programming Reference, July 1990, http://www.w3.org/Graphics/GIF/spec-gif89a.txt

[IEEE754] IEEE, "IEEE Standard for Binary Floating-Point Arithmetic", IEEE 754-1985, October 1985, http://ieeexplore.ieee.org/servlet/opac?punumber=2355

[ISO-15924] International Organization for Standardization, "ISO 15924 Registration Authority", http://www.unicode.org/iso15924/

[ISO-8601] International Organization for Standardization, "Data Elements and Interchange Formats - Information Interchange - Representation of Dates and Times", ISO/IEC 8601:2004, December 2004, http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=40874&ICS1=1&ICS2=140&ICS3=30

Note There is a charge to download the specification.

[ISO/IEC29500-1:2016] ISO/IEC, "Information technology -- Document description and processing languages -- Office Open XML File Formats -- Part 1: Fundamentals and Markup Language Reference", ISO/IEC 29500-1:2016, https://www.iso.org/standard/71691.html

[ISO/IEC29500-2:2012] ISO/IEC, "Information technology -- Document description and processing languages -- Office Open XML File Formats -- Part 2: Open Packaging Conventions", ISO/IEC 29500-2:2012, http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=61796

[ISO/IEC29500-3:2015] ISO/IEC, "Information technology -- Document description and processing languages -- Office Open XML File Formats -- Part 3: Markup Compatibility and Extensibility", https://www.iso.org/standard/65533.html

[JFIF] Hamilton, E., "JPEG File Interchange Format, Version 1.02", September 1992, http://www.w3.org/Graphics/JPEG/jfif.txt

[MS-EMF] Microsoft Corporation, "Enhanced Metafile Format".

[MS-OAUT] Microsoft Corporation, "OLE Automation Protocol".

[MS-ODBCSTR] Microsoft Corporation, "ODBC Connection String Structure".

[MSDN-BMPST] Microsoft Corporation, "Bitmap Storage", http://msdn.microsoft.com/en-us/library/dd183391(VS.85).aspx

[RFC2083] Boutell, T., et al., "PNG (Portable Network Graphics) Specification Version 1.0", RFC 2083, March 1997, https://www.rfc-editor.org/info/rfc2083

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, https://www.rfc-editor.org/info/rfc2119

[RFC3302] Parsons, G., and Rafferty, J., "Tag Image File Format (TIFF) - image/tiff MIME Sub-Type Registration", RFC 3302, September 2002, https://www.rfc-editor.org/info/rfc3302

[RFC3629] Yergeau, F., "UTF-8, A Transformation Format of ISO 10646", STD 63, RFC 3629, November 2003, https://www.rfc-editor.org/info/rfc3629

[RFC4646] Phillips, A., and Davis, M., Eds., "Tags for Identifying Languages", BCP 47, RFC 4646, September 2006, https://www.rfc-editor.org/info/rfc4646

[RFC4647] Phillips, A., and Davis, M., Eds., "Matching of Language Tags", BCP 47, RFC 4647, September 2006, http://www.rfc-editor.org/rfc/rfc4647.txt

[RFC5234] Crocker, D., Ed., and Overell, P., "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008, https://www.rfc-editor.org/info/rfc5234

[XMLSCHEMA1] Thompson, H., Beech, D., Maloney, M., and Mendelsohn, N., Eds., "XML Schema Part 1: Structures", W3C Recommendation, May 2001, https://www.w3.org/TR/2001/REC-xmlschema-1-20010502/

[XMLSCHEMA2] Biron, P.V., Ed. and Malhotra, A., Ed., "XML Schema Part 2: Datatypes", W3C Recommendation, May 2001, https://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

1.2.2 Informative References

[MS-OLEDS] Microsoft Corporation, "Object Linking and Embedding (OLE) Data Structures".

[MSDN-CompareOptions] Microsoft Corporation, "CompareOptions Enum", https://learn.microsoft.com/en-us/dotnet/api/system.globalization.compareoptions

[MSDN-ENCLOC] Microsoft Corporation, "Encoding and Localization", .NET Framework Developer's Guide, http://msdn.microsoft.com/en-us/library/h6270d0z.aspx

[MSDN-ToDouble] Microsoft Corporation, "Convert.ToDouble Method", .NET Framework Class Library, http://msdn.microsoft.com/en-us/library/system.convert.todouble.aspx

[RFC2616] Fielding, R., Gettys, J., Mogul, J., et al., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999, https://www.rfc-editor.org/info/rfc2616

1.3 Overview

This structure describes a ZIP archive that stores all the information needed to describe a web drawing.

A Document XML Part in the ZIP archive describes the properties of the web drawing.

A collection of Visio parts and Shared XML parts in the ZIP archive describes the graphical elements displayed in the web drawing. These graphical elements are presented as Shapes on Drawing Pages. Shapes are described by the Master XML Part, Page XML Part, and Themes XML Part. Drawing Pages are described by the Masters XML Part and Pages XML Part.

Graphical elements can be static or dynamic. Dynamic graphical elements have visual properties that are bound to data in a **data source**, and the appearance of these elements changes as data in the data source refreshes (section 2.2.10). A collection of Visio parts in the ZIP archive describes the Data Connections, bindings (section 2.2.10.2.1) between data and shapes, and recalculation information necessary to update (section 2.2.11) visual properties. Data connections are described by the Connections XML part. Data bindings are described by the Recordsets XML part. Recalculation information is described by a grammar (section 2.2.11.2.1) for Formula Evaluation that describes how changes in the data are translated into changes in properties of graphical elements. This grammar is described by the Master XML part and Page XML part.

Additional items in the ZIP archive describe the Images and Comments in the web drawing.

1.4 Relationship to Protocols and Other Structures

This specification is dependent on the structures and concepts defined in [ISO/IEC29500-2:2012], [ISO/IEC29500-3:2015] and [ISO/IEC29500-1:2016] section 9 for Open Packaging Conventions.

1.5 Applicability Statement

This document specifies a persistence format for Web Drawing content, which can include Drawing Pages, Masters, Shapes, Images, Comments, Data Connections, and recalculation information, as specified in Section 2.2.1. The persistence format is applicable when the document content is graphical in nature.

This persistence format is applicable for use as a stand-alone document, and for containment within other documents as an **embedded object**, as described in [MS-OLEDS].

This persistence format provides interoperability with applications that create or read documents conforming to this structure.

1.6 Versioning and Localization

This document covers versioning issues in the following areas:

- App XML Part
- Custom XML Part
- Version

This document covers localization in the Core XML Part.

Local overrides to document language are specified in attributes, properties, and function arguments as described in the Conceptual Overview, Visio XML Schema, ShapeSheet Properties, and Formula Expressions and Evaluation sections.

1.7 Vendor-Extensible Fields

Persistence format can be extended by storing information in Parts that are not specified in Section 2. Implementations are not required to preserve or remove additional Parts when modifying an existing document.

2 Structures

This section specifies the overall structure of a file that conforms to this specification.

2.1 File Structure Overview

A file of the type specified by this specification MUST be a Package that is a ZIP archive.

The ZIP Package is used to persist information that is necessary to fully represent a web Drawing. This package contains a collection of Parts that are used to persist data in XML or standard binary formats, and to specify various aspects of the Web Drawing as well as the structure of the Package.

2.1.1 Package

A file of the type specified by this document MUST be a Package that is a ZIP archive and that conforms to the Open Packaging Conventions as specified in [ISO/IEC29500-2:2012], the further packaging restrictions specified in [ISO/IEC29500-1:2016] section 9, and this specification.

2.1.2 Parts

A Package is composed of multiple parts as specified in [ISO/IEC29500-2:2012] section 9.1. Each part has an associated content type that specifies the format it is persisted in. Each part can also be the target or the source of a connection between two parts called a relationship (section 2.1.3), as specified in [ISO/IEC29500-2:2012] section 9.3.

The valid parts, content types, required relationships, and optional relationships between all parts in this package are specified in Part Enumeration.

2.1.3 Relationship

A relationship specifies a connection between a source and a target resource as specified in [ISO/IEC29500-1:2016] section 9.2. Relationship identifiers are used in binary and XML part (section 2.1.2) content to reference unique relationship elements in relationship parts that in turn target other resources.

There are several different types of relationships:

- A Package relationship is a relationship where the target is a part and the source is the package as a whole.
- A part-to-part relationship is a relationship where the target is a part and the source is a part in the package.
- An explicit relationship is a relationship where a resource is referenced from the contents of a source part by referencing the **ID** attribute value of a relationship element.
- An implicit relationship is a relationship where a resource is not referenced from the contents of a source part by referencing the **ID** attribute value of a relationship element.
- An internal relationship is a relationship where the target is a part in the package.
- An external relationship is a relationship where the target is an external resource, not part of the package.

2.1.4 Markup Compatibility

A markup specification defines a set of elements and attributes within one or more namespaces. A characteristic of an application that consumes the markup is that it can recognize the elements and attributes within understood namespaces, including those containing elements and attributes defined in the markup specification. Markup consumers MUST treat all recognized elements and attributes of any understood namespace according to the requirements of the markup specifications defining those elements or attributes. A markup specification MAY require that the presence of unrecognized elements or attributes in an understood namespace be treated as an error condition; however, markup consumers MUST treat the presence of an unrecognized element or attribute from the Markup Compatibility namespace as an error condition.

If a markup consumer encounters an element or attribute from a non-understood namespace, the markup consumer MUST treat the presence of that element or attribute as an error condition, unless the markup producer has embedded in the markup document explicit Markup Compatibility elements or attributes that override that behavior.

The valid Markup Compatibility elements and attributes in a Web Drawing are specified in the Markup Compatibility Schema (section 2.3.5).

2.2 Conceptual Overview

The Conceptual Overview sections that follow specify how higher-level features of the file format are represented by combinations of parts and XML elements.

2.2.1 Web Drawing

A web drawing is a collection of Drawing Pages, Masters, Shapes, Images, Comments, Data Connections, and recalculation information that can be rendered as a **drawing** in a web browser.

A web drawing is specified by a Package as specified in the File Structure Overview. The contents of a web drawing are specified by the Parts in the Part Enumeration section.

For examples of various web drawings, see Structure Examples.

2.2.2 Drawing Page

A drawing page is a collection of Shapes that are viewed together.

A collection of drawing pages in a web drawing is specified by a Pages XML Part.

2.2.2.1 Page Identification

A Page_Type element in a Pages XML Part specifies a single drawing page. A drawing page is uniquely identified by the **ID**, **Name**, and **NameU** attributes in a Page_Type element. The following elements in parts of the web drawing have attributes that are equal to **ID**, **Name**, or **NameU** and specify supplementary information about the drawing page.

- A PublishedPage_Type element in a Document XML Part has an **ID** attribute that is equal to the **ID** attribute of the Page_Type element, and specifies that the drawing page is viewable in the web drawing.
- A **TitlesOfParts** element in an App XML Part contains an **lpstr** element with contents equal to the **Name** attribute of the Page_Type element and specifies the name of the drawing page.
- A Page_Type element in a Pages XML Part can have a **BackPage** attribute that is equal to the **ID** attribute of the Page_Type element, and specifies that the latter drawing page is to be used as the background page for the former drawing page.

- A RowMap_Type element in the Recordsets XML Part contains a **PageID** attribute that is equal to the **ID** attribute of the Page_Type element, and specifies the data binding between a **row** of a Recordset and a Shape on the drawing page.

The graphical information necessary to render a drawing page is specified by the PageSheet_Type and ShapeSheet_Type elements in a Pages XML Part and a Shapes_Type element in a Page XML Part.

A drawing page is also associated with a Master. The graphical information about a Master is specified by the PageSheet_Type and Shapes_Type elements in a Master XML Part.

A drawing page can contain **embedded images**. Each Image used in a drawing page is specified by an Image Part. The Fallback Image section explains how some embedded image formats and **embedded objects** are rendered using Fallback Images which are also specified by Image Parts.

2.2.2.2 Coordinate System

A point on a drawing page is specified by coordinates on a two-dimensional Cartesian plane, where the x-coordinate specifies the horizontal position and the y-coordinate specifies the vertical position.

The origin of a drawing page is the lower-left corner of the drawing page.

Increasing the x-coordinate specifies the position of a Shape, group or object rightward, while increasing the y-coordinate specifies the position upward.

Every drawing page defines its own coordinate system.

2.2.2.3 Drawing Scale

The drawing scale of a drawing page is the ratio of the values of the PageScale Cell_Type element to the value of the DrawingScale Cell_Type element.

Drawing units specify size or position of objects on the drawing page. Page units specify measurements on the printed page.

The drawing scale multiplied by the drawing units will result in a scaled object.

The following cells are not expressed in drawing units and are not scaled. All other vLengths are expressed in drawing units and will be scaled.

- BeginArrowSize, EndArrowSize
- GlowSize
- ReflectionBlur
- ReflectionDist
- SoftEdgesSize
- LineWeight
- FontScale, Size
- AsianFont
- Case
- Color, ColorTrans
- DblUnderline

- ComplexScriptFont, ComplexScriptSize
- DoubleStrikethrough

- Overline
- Pos
- Strikethru
- Style
- BevelTopWidth, BevelTopHeight, BevelBottomWidth, BevelBottomHeight
- BevelDepthSize, BevelContourSize
- ShdwOffsetX, ShdwOffsetY
- DistanceFromGround

2.2.2.4 Foreground Page

A drawing page can be a foreground page. A web drawing contains at least one foreground page.

The Page_Type element specifies whether a page is a foreground page in a web drawing. If a Page_Type element in a Pages XML Part contains a **Background** attribute equal to zero, it is a foreground page.

A foreground page in a web drawing has zero or one background pages as specified by the **BackPage** attribute of the Page_Type element associated with the page.

A foreground page can be published or unpublished. Published pages are viewable in a web drawing while unpublished pages are hidden. The PublishSettings_Type child element of the VisioDocument_Type element for the web drawing determines whether a page is published or unpublished.

2.2.2.5 Background Page

A background page is a drawing page that can appear behind foreground pages and other background pages in a web drawing.

A background page can have a different drawing scale than a foreground page.

A background page in a web drawing is specified by the Page_Type element associated with the page.

If the Page_Type element associated with the page contains a **Background** attribute equal to one, it is a background page.

A background page in a web drawing has zero or one background pages as specified by the **BackPage** attribute of the Page_Type element associated with the page.

2.2.2.6 Layer

A web drawing can have layers. A shape belongs to zero or more layers. A layer can contain zero or more shapes. A layer specifies additional information about the shapes that it contains such as color, color transparency, and visibility.

A layer in a web drawing is specified by the Row_Type child element of a Layer Section_Type element. A Layer Section_Type element is a child of a PageSheet_Type element associated with the page.

Each Row_Type child element of the Layer Section_Type element contains information for a single layer. A layer is uniquely identified by the **IX** attribute of that layer's Row_Type.

A collection of Cell_Type elements that define a layer's properties is composed of Color, Visible, Lock, and ColorTrans.

The layer membership of a shape is specified by the LayerMember Cell_Type element in the ShapeSheet_Type element of the shape.

2.2.3 Shape

A shape is a collection of Geometry Visualization, Format, Text, Images, and Shape Data in a Drawing Page.

2.2.3.1 Shape Identification

A Shape in a Web Drawing is specified by a ShapeSheet_Type child element of a Shapes_Type descendant element of either a PageContents element in a Page_XML_Part, or a MasterContents element in a Master XML Part.

A Shape is uniquely identified within a Drawing Page by the **ID** attribute of its ShapeSheet_Type element. The following elements in other Parts of the document have attributes that reference shapes by their **ID** attributes to specify supplementary information about them.

- A RowMap_Type element specifies the shape it is bound to in its **ShapeID** attribute.
- A CommentEntry_Type element specifies the shape it relates to in its **ShapeID** attribute.
- The ShapeSheetRef Reference Token references a shape.

2.2.3.1.1 One-Dimensional Shape

A Shape is one-dimensional if its ShapeSheet_Type element has BeginX, BeginY, EndX, and EndY child elements of the type Cell_Type.

2.2.3.1.2 Two-Dimensional Shape

A Shape is two-dimensional if its ShapeSheet_Type element has no BeginX, BeginY, EndX, or EndY child elements of the type Cell_Type.

2.2.3.2 Geometry Visualization

Geometry on Shapes in a Web Drawing can be visualized.

The following sections specify the concepts and elements of geometry visualization.

2.2.3.2.1 Coordinate System

A point on a Drawing Page or a Shape is specified by coordinates on a two-dimensional Cartesian plane, where the x-coordinate specifies the horizontal position and the y-coordinate specifies the vertical position.

Every Shape defines a local coordinate system. A point on a shape is specified either in its local coordinates or in the coordinate system of the shape's Parent, depending on the **N** attribute of the Cell_Type element specifying this point.

A point specified in local coordinates can be converted into parent coordinates by applying the following transformations in the following order:

1. Subtract the value of the LocPinX property of the Cell_Type element from the x-coordinate.
2. Subtract the value of the LocPinY property of the Cell_Type from the y-coordinate.
3. Mirror the point about the y-axis if the value of the FlipX property of the Cell_Type is equal to one.
4. Mirror the point about the x-axis if the value of the FlipY property of the Cell_Type is equal to one.
5. Rotate the point counterclockwise around the origin by the value of the Angle property of the Cell_Type.
6. Add the value of the PinX Cell_Type to the x-coordinate.
7. Add the value of the PinY Cell_Type to the y-coordinate.

2.2.3.2.1.1 Relative Coordinate System

A relative coordinate system is a Coordinate System where the coordinates are determined by multiplying a scalar value by the width or height of the Shape.

It is used to represent x-coordinate or y-coordinate by the Cell_Type element that has a RelCubBezTo, RelEllipticalArcTo, RelLineTo, RelMoveTo or RelQuadBezTo properties of the Row_Type parent element.

It is also used to represent formula by E Cell_Type element that has a NURBSTo Row_Type parent element and A Cell_Type element that has a PolylineTo Row_Type parent element.

The width and height are specified by the Width and Height Cell_Type elements.

2.2.3.2.2 Geometry Path

A path is a collection of vertices and line or curve segments that specifies an enclosed area. The geometry of a shape is specified by a collection of paths.

Each Geometry Section_Type element specifies a path. Each Row_Type child element specifies a vertex of that path, a segment of that path, or both.

If the Row_Type element is of type Ellipse or InfiniteLine, it specifies the only segment of the path.

Otherwise, if the Row_Type element is of type MoveTo or RelMoveTo, it specifies the first vertex in the path or the first vertex after a break in the path.

Otherwise, the Row_Type element specifies a vertex and a segment that connects the vertex of the previous Row_Type element to the vertex specified in the current Row_Type element.

For a path to be visible, the following conditions are necessary.

- The shape containing the path is not on a layer whose Visible Cell_Type element has a value equal to zero.
- The value of the NoShow Cell_Type child of the path's Geometry Section_Type element is not equal to one.

The visibility of the path's line and the visibility of the path's fill are specified, respectively, by the NoLine and NoFill Cell_Type child elements of the path's Geometry Section_Type element.

The format of the path's line and the format of the path's fill are specified, respectively, by the line property and fill property of the shape containing the path.

2.2.3.2.3 Display Order

The display order of shapes in a drawing page is a strict ordering that is used to determine their visualization behavior. If one shape has a higher position than another in the display order, the former shape is displayed in front of the latter.

If one ShapeSheet_Type element appears before another ShapeSheet_Type element in the same XML Part, the shape specified by the latter ShapeSheet_Type element has a higher position in the display order.

If one shape is a member shape of another shape, the relative positions of the two shapes in the display order are specified by the DisplayMode Cell_Type of the latter shape.

2.2.3.3 Shape Hierarchy

Shapes can be hierarchically grouped. A shape contains zero or more subshapes.

2.2.3.3.1 Parent

If a ShapeSheet_Type element has a parent Shapes_Type element whose parent is a ShapeSheet_Type element, the shape specified by the latter ShapeSheet_Type element is called that shape's parent.

If a ShapeSheet_Type element has an ancestor Shapes_Type element whose parent is a ShapeSheet_Type element, the shape specified by the latter ShapeSheet_Type element is called an ancestor shape of the shape specified by the former ShapeSheet_Type element.

2.2.3.3.2 Top-Level Shape

Top-level shapes are ShapeSheet_Type elements that have no ShapeSheet_Type ancestors. The parent of top-level shapes is the drawing page.

2.2.3.3.3 Subshape

If a ShapeSheet_Type element has a parent Shapes_Type element whose parent is a ShapeSheet_Type element, the shape specified by the former ShapeSheet_Type element is called a subshape of the shape specified by the latter ShapeSheet_Type element.

If a ShapeSheet_Type element has an ancestor Shapes_Type element whose parent is a ShapeSheet_Type element, the shape specified by the former ShapeSheet_Type element is called a member shape of the shape specified by the latter ShapeSheet_Type element.

2.2.3.4 Shape Selection

Shapes in a Web drawing can be selected.

For a shape to be selectable, all the following conditions are necessary.

- The value of the LockSelect Cell_Type element of the shape is equal to zero, or value of the ProtectShapes_Type element of the shape is equal to zero.
- The shape is not on a layer whose Visible Cell_Type element has a value equal to zero.
- The shape is not on a layer whose Lock Cell_Type element has a value equal to zero.
- None of the ancestor shapes of the shape has a SelectMode cell whose value is equal to zero.
- The shape has at least one visible geometry path that is not obscured by shapes with a higher display order.
- The shape is on a foreground page.

2.2.3.5 Shape Hyperlinks

A shape has zero or more hyperlinks associated with it. Hyperlinks point to drawing pages within the Web drawing, shapes within the Web drawing, or destinations outside the Web drawing.

The set of hyperlinks associated with a shape is specified by the Hyperlink Section_Type element.

Each hyperlink is specified by a Row_Type child element of the Hyperlink Section_Type element for the shape. This Row_Type element specifies the information about the hyperlink properties using a collection of Cell_Type elements. It is either contained under a ShapeSheet_Type element for the shape or inherited.

A collection of Cell_Type elements that define the properties of the hyperlink is composed of Description, Address, SubAddress, ExtraInfo, Default, Invisible, and SortKey Cell_Type elements.

2.2.3.6 Shape Data

A shape can have data associated with it that provides information about its meaning. A shape's data is stored as a set of shape data **fields**.

Each shape data field is specified by a Row_Type child element of the Property Section_Type element for the shape. This Row_Type element specifies the information about the shape data field properties using a collection of Cell_Type elements. It is either contained under a ShapeSheet_Type element for the shape or it is inherited.

A collection of Cell_Type elements that define the properties of the shape data field is composed of Calendar, DataLinked, Format, Invisible, Label, LangID, Type, and Value Cell_Type elements.

The name of a shape data field is specified by the **N** attribute of the Row_Type element for the field. The value of a shape data field is specified by the Value Cell_Type element. The data type of a shape data field is specified by the Type Cell_Type element.

2.2.4 Master

Masters specify shapes that can be reused throughout a web drawing.

A shape on a drawing page can be linked to a master, which can affect various properties of the shape including its visual appearance. A relationship to such a master is called master-to-shape inheritance.

2.2.4.1 Master Identification

A master is specified by the combination of a Master_Type element in a Masters XML Part, and the ShapeSheet_Type elements in the Master XML Part specified by the Master_Type element's Rel_Type child element. These ShapeSheet_Type elements are called master shapes.

The following elements in other parts of the document have attributes that reference masters.

- A ShapeSheet_Type element in a Page XML Part can specify with its **Master** attribute the master it inherits from.
- The Use function token accepts as its argument the name or **GUID** of a master.
- The MasterSheetRef reference token references a master.

2.2.5 Sheet

A sheet is a collection of properties that specify information for a shape, master, drawing page, style, or web drawing.

2.2.5.1 Sheet Identification

A sheet for a shape is a collection of sections, rows, and cells contained in a ShapeSheet_Type element in a Page XML Part. A sheet for a shape is uniquely identified by the **ID** attribute in a Shapes_Type element.

A sheet for a master is a collection of sections, rows, and cells contained in a ShapeSheet_Type element in a Master XML Part. A sheet for a master is uniquely identified by the **UniqueID** attribute in a Shapes_Type element.

A sheet for a drawing page is a collection of sections, rows, and cells contained in a PageSheet_Type element in a Masters XML Part or Pages XML Part. A sheet for a drawing page is uniquely identified by the **ID**, **Name,** and **NameU** attributes in a Pages_Type element. A sheet for a drawing page is unique in a Master_Type or Page_Type element.

A sheet for a style is a collection of sections, rows, and cells contained in a StyleSheet_Type element in a Document XML Part. A sheet for a style is uniquely identified by the **ID** attribute in a StyleSheets_Type element.

A sheet for a web drawing is a collection of sections, rows, and cells contained in a DocumentSheet_Type element in a Document XML Part. A sheet for a Web drawing is unique in a VisioDocument_Type element.

2.2.5.2 Sheet Types

A sheet is specified by a Sheet_Type abstract complex type. A sheet in a web drawing can be one of four distinct types that extend the Sheet_Type. The distinct types are shape sheet, page sheet, style sheet, and document sheet (section 2.2.5.2.1).

2.2.5.2.1 Document Sheet

A document sheet specifies information pertaining to a web drawing. It is a collection of sections, rows, and cells in a DocumentSheet_Type child element of the VisioDocument_Type element in the Document XML Part.

2.2.5.2.2 Page Sheet

A page sheet specifies information pertaining to a drawing page. It is a collection of sections, rows, and cells contained in a Pages XML Part or Masters XML Part. Each page sheet is specified by a PageSheet_Type child element of a Page_Type child element of a Pages_Type element in either a Pages XML Part or PageSheet_Type child element of a Master_Type child element of a Masters_Type element in a Masters XML Part.

2.2.5.2.3 Shape Sheet

A shape sheet specifies information pertaining to a shape or master.

A shape sheet pertaining to a shape in a web drawing is a collection of sections, rows, and cells contained in a Page_XML_Part. Each shape sheet is specified by a ShapeSheet_Type child element of a Shapes_Type descendant element of a PageContents element in a part.

A shape sheet pertaining to a master in a web drawing is a collection of sections, rows, and cells contained in a Master XML Part. Each shape sheet is specified by a ShapeSheet_Type child element of a Shapes_Type descendant element of a MasterContents element in a part.

2.2.5.2.4 Style Sheet

A style sheet specifies information pertaining to a style and is used in inheritance.

A style sheet in a web drawing is a collection of sections, rows, and cells contained in a Document XML Part. Each style sheet is specified by a StyleSheet_Type child element of the StyleSheets_Type child element of the VisioDocument element in a part.

2.2.5.2.4.1 Root Style Sheet

The root style sheet is a style sheet in a web drawing that all other style sheets inherit from.

The root style sheet is specified by the StyleSheet_Type element whose **ID** attribute value is equal to zero and whose **NameU** attribute value is equal to "No Style".

2.2.5.3 Sheet Structures

A sheet structure is where the property information of a sheet has been hierarchically grouped into sections, rows, and cells.

2.2.5.3.1 Section

A section specifies a collection of related properties of a sheet. A section contains cells and rows.

Sections are specified by Section_Type child elements of the ShapeSheet_Type, PageSheet_Type, StyleSheet_Type, and DocumentSheet_Type elements. The **N** attribute of a Section_Type element specifies the name of the section that identifies the collection of properties that it pertains to. The properties specified by a section are specified by the Cell_Type and Row_Type child elements of the Section_Type element.

2.2.5.3.2 Row

A row specifies a subset of the properties in a section. A row contains cells.

Rows are specified by Row_Type child elements of the Section_Type child elements of the ShapeSheet_Type, PageSheet_Type, StyleSheet_Type, and DocumentSheet_Type elements. The **N** attribute of a Row_Type element specifies the name of the row that identifies the subset of properties that it pertains to. The properties specified by a row are specified by the Cell_Type child elements of the Row_Type element.

2.2.5.3.3 Cell

A cell specifies a single property in a row, section, or sheet.

Cells are specified by Cell_Type child elements of the Section_Type, Row_Type, ShapeSheet_Type, PageSheet_Type, StyleSheet_Type, and DocumentSheet_Type elements. The **N** attribute of a Cell_Type element specifies the name of the cell that identifies the property that it pertains to.

The **V** attribute of a Cell_Type element specifies the value of the property of the cell. The **F** attribute of a Cell_Type element specifies the formula expression of the property of the cell.

If the **F** attribute is present, the value of the property is used until it is replaced by a value from the most recent formula evaluation that does not result in an error value.

2.2.5.3.3.1 Cell Default Values

The property value assigned to a missing or malformed cell is called a cell default value. If the Cell_Type element of a cell in a web drawing is not specified directly in a sheet or through inheritance, the cell is called a missing cell. If the Cell_Type element of a cell in a web drawing does not specify a **V** attribute, the cell is called a malformed cell.

The cell default value of a missing cell depends on its parse token, custom structure, or custom token grouping. The cell default value for parse tokens and custom structures is specified in the following table. When the default value for all the parse tokens in a custom token grouping is the same, the custom token grouping is specified in the table instead of individual parse tokens.

Parse token, custom structure, or custom token grouping	Cell default value
PtgNum	0.00
PtgBool	0
PtgString	""
PtgByte	0
PtgColorRGB	#000000
PtgShort	0
PtgDate	0.00 days
PtgInt	0
PtgUnsShort	0
PtgNumI	0.00 inches
vLanguageString	""
vFont	"0"
vAny	0.00 days
vAngle	0.00 radians
vLength	0.00 inches
vColor	#000000
vFormatString	""

The cell default value of a malformed cell depends on the **U** attribute value of its Cell_Type element. If the **U** attribute of the Cell_Type element of a malformed cell is not specified, the cell default value is specified in the previous table.

If the **U** attribute of the Cell_Type element of a malformed cell is specified, the cell default value is specified in the following table.

U attribute value	Cell default value
AC	0.00 inches
DEG	0.00 radians
DA	0.00 radians
AD	0.00 radians
RAD	0.00 radians
BOOL	0
COLOR	#000000

U attribute value	Cell default value
CY	0.00
DATE	0.00
ED	0.00 days
EH	0.00 days
EM	0.00 days
ES	0.00 days
EW	0.00 days
HA	0.00 inches
CM	0.00 inches
DL	0.00 inches
FT	0.00 inches
F_I	0.00 inches
IN	0.00 inches
IN_F	0.00 inches
KM	0.00 inches
M	0.00 inches
MI	0.00 inches
MI_F	0.00 inches
MM	0.00 inches
NM	0.00 inches
PER	0.00
YD	0.00 inches
DP	0.00 inches
PNT	PNT(0.00, 0.00)
STR	""
DE	0.00 days
C_D	0.00 inches
C	0.00 inches
D	0.00 inches
DT	0.00 inches
P	0.00 inches
P_PT	0.00 inches

U attribute value	Cell default value
PT	0.20 inches

2.2.5.4 Inheritance

This section describes how properties are inherited in sheets in a web drawing.

2.2.5.4.1 Master-to-Shape Inheritance

A shape on a drawing page can be linked to a master, which can affect various properties of the shape including its visual appearance. A relationship to such a master is called master-to-shape inheritance, and the shape is called an instance of that master. A shape has zero or one master-to-shape inheritance relationships.

If the **Master** attribute of the ShapeSheet_Type element of a shape on a drawing page is equal to the **ID** attribute of a Master_Type element of a master, the shape is an instance of the master. Any sections, rows, cells, or subshapes not specified in the instance are inherited from the master.

An instance can modify the sections, rows, and cells taken on from inheritance by specifying local properties. In addition, if an instance contains a subshape whose ShapeSheet_Type element has a **MasterShape** attribute that matches the **ID** attribute of a subshape of the master, the local properties specified in this subshape will override those of the corresponding subshape in the master.

If a master has one top-level shape, a shape that inherits from that master inherits the descendant elements of that master shape. If a master has more than one master shape, a shape that inherits from that master inherits those master shapes as subshapes.

2.2.5.4.2 Style-to-Shape Inheritance

Shapes in a web drawing can be linked to a style, which can affect various properties of the shape including its visual appearance. A relationship to such a style is called style-to-shape inheritance. A style-to-shape inheritance allows a shape to take on properties from the style it inherits from. A shape can have zero to three style-to-shape inheritance relationships.

The style-to-shape inheritances in a web drawing are specified by the Page XML Part. Each style-to-shape inheritance is specified by the attributes of a ShapeSheet_Type child element of the Shapes_Type descendant element of the PageContents element in a part. Style-to-shape inheritance information is specified by the ShapeSheet_Type element and a StyleSheet_Type child element of a StyleSheets_Type child element of the VisioDocument element in the Document XML Part.

If the **LineStyle**, **FillStyle**, and **TextStyle** attributes of the ShapeSheet_Type element are empty, a style-to-shape inheritance is not specified. If the **LineStyle**, **FillStyle**, or **TextStyle** attributes of the ShapeSheet_Type element are not empty, a style-to-shape inheritance exists individually for each attribute between the ShapeSheet_Type element and the StyleSheet_Type element whose **ID** attribute value is equal to the value of a **LineStyle**, **FillStyle**, or **TextStyle** attribute of the ShapeSheet_Type element.

The **LineStyle**, **FillStyle**, and **TextStyle** attributes of a ShapeSheet_Type element each specify a set of Cell_Type child elements of the StyleSheet_Type element as specified in the following table.

Attribute	Cell_Type elements
LineStyle	Specifies Cell_Type elements related to line properties except for Cell_Type child elements of a FillGradient Section_Type.

Attribute	Cell_Type elements
FillStyle	Specifies Cell_Type elements related to fill properties and effect properties including Cell_Type child elements of a FillGradient Section_Type.
TextStyle	Specifies Cell_Type elements related to text.

2.2.5.4.3 Style-to-Master Inheritance

Masters in a web drawing can be linked to a style, which can affect various properties of the master including its visual appearance. A relationship to such a style is called style-to-master inheritance. A style-to-master inheritance allows a master to take on properties from the style it inherits from. A master can have zero to three style-to-master inheritance relationships.

The style-to-master inheritances in a web drawing are specified by the Master XML Part. Each style-to-master inheritance is specified by the attributes of a ShapeSheet_Type child element of the Shapes_Type descendant element of the MasterContents element in a part. Style-to-master inheritance information is specified by the ShapeSheet_Type element and a StyleSheet_Type child element of a StyleSheets_Type child element of the VisioDocument element in the Document XML Part.

If the **LineStyle**, **FillStyle**, and **TextStyle** attributes of the ShapeSheet_Type element are empty, a style-to-master inheritance is not specified. If the **LineStyle**, **FillStyle**, or **TextStyle** attributes of the ShapeSheet_Type element are not empty, a style-to-master inheritance exists individually for each attribute between the ShapeSheet_Type element and the StyleSheet_Type element whose **ID** attribute value is equal to the value of a **LineStyle**, **FillStyle**, or **TextStyle** attribute of the ShapeSheet_Type element.

The **LineStyle**, **FillStyle**, and **TextStyle** attributes of a ShapeSheet_Type element each specify a set of Cell_Type child elements of the StyleSheet_Type element as specified in the table found in section 2.2.5.4.2.

2.2.5.4.4 Style-to-Style Inheritance

Styles in a web drawing can be linked to other styles, which can affect various properties of the style. A relationship to such a style is called style-to-style inheritance. A style-to-style inheritance allows a style sheet to take on properties from the style it inherits from. A style can have zero to three style-to-style inheritance relationships.

The style-to-style inheritances in a web drawing are specified by the Document XML Part. Each style-to-style inheritance is specified by the attributes of a StyleSheet_Type child element of the StyleSheets_Type child element of the VisioDocument element in a part. Style-to-style inheritance information is specified by the StyleSheet_Type element and other StyleSheet_Type elements in the Document XML Part.

If the **LineStyle**, **FillStyle**, and **TextStyle** attributes of the StyleSheet_Type element are empty, a style-to-style inheritance is not specified. If the **LineStyle**, **FillStyle**, or **TextStyle** attributes of the StyleSheet_Type element are not empty, a style-to-style inheritance exists individually for each attribute between the StyleSheet_Type element and another StyleSheet_Type element whose **ID** attribute value is equal to the value of a **LineStyle**, **FillStyle**, or **TextStyle** attribute of the StyleSheet_Type element.

The **LineStyle**, **FillStyle**, and **TextStyle** attributes of a StyleSheet_Type element each specify a set of Cell_Type child elements of the StyleSheet_Type element as specified in the table found in section 2.2.5.4.2.

2.2.5.4.5 Theme Inheritance

A shape in a web drawing takes on the format properties specified by its dynamic theme components and quick style slices through inheritance. Format properties from the dynamic theme are represented as a dynamic theme style sheet that is linked to from other style sheets, masters, and shapes. A relationship to a dynamic theme style sheet is called theme inheritance.

Theme inheritance allows a style sheet, master, or shape to take on properties from the cells of the dynamic theme style sheet it inherits from. These inheritances in a web drawing are specified as style-to-shape inheritance, style-to-master inheritance, and style-to-style inheritance.

A dynamic theme style sheet in a web drawing is specified by a StyleSheet_Type child element of the StyleSheets_Type child element of the VisioDocument element in the Document XML Part. A dynamic theme style sheet is uniquely identified by a StyleSheet_Type element whose **NameU** attribute is equal to "Theme".

2.2.5.4.6 Local Properties

Sheets corresponding to styles, masters, and shapes in a web drawing can specify that their own properties replace properties taken on from inheritance. These properties are called local properties.

Local properties are specified by Cell_Type, Row_Type, or Section_Type descendant elements of Sheet_Type elements. A local property replaces the properties of an inherited Cell_Type, Row_Type, or Section_Type element, if the value of the local property's **N** attribute is equal to the value of the **N** attribute of the inherited Cell_Type, Row_Type, or Section_Type element.

2.2.5.5 Sheet Extensibility

Sheet extensibility is a mechanism whereby a web drawing specifies extensions to the rules about sections, rows, cells, and function tokens as defined in this specification. Such extensions are specified in SectionDef_Type, RowDef_Type, CellDef_Type, and FunctionDef_Type descendant elements of the Extensions element of the Extensions XML Part.

The valid **N** attributes of a Section_Type element are specified in section 2.4.1. However, SectionDef_Type elements can specify additional valid **N** attributes.

The valid **N** attributes of a Cell_Type element and the locations where a Cell_Type element with a given **N** attribute can occur are specified in section 2.4.4. However, CellDef_Type elements can specify additional valid **N** attributes. Additionally, a CellDef_Type element specifies the valid locations where a CellDef_Type element with a given **N** attribute can occur, based on the CellDef_Type element's ancestor SectionDef_Type and RowDef_Type elements. Cells defined through sheet extensibility are used for formula evaluation only.

The valid function tokens are specified in section 2.5.3. However, FunctionDef_Type elements can specify additional valid function tokens. A function token defined through sheet extensibility consumes all argument, and returns a PtgErr parse token with an error code equal to #VALUE!.

2.2.6 Image

A web drawing can have **embedded images**. Each embedded image is associated with a shape, which provides information about the image placement, size, and properties.

The ShapeSheet_Type element of a shape that specifies an image MUST have its **Type** attribute equal to "Foreign" and MUST have a ForeignData_Type child element.

The ShapeSheet_Type element specifies the position, width, and height of the image using the ImgOffsetX, ImgOffsetY, ImgWidth, and ImgHeight Cell_Type child elements.

The ShapeSheet_Type element specifies image formatting properties using the Blur, Brightness, Contrast, Denoise, Gamma, and Sharpen, and Transparency Cell_Type child elements.

Additional image properties, such as format and compression, are specified by attributes of the ForeignData_Type child element. This element MUST have a Rel_Type child element that specifies an explicit relationship to another Part. The following embedded image formats are supported:

- **bitmap (BMP)**
- **enhanced metafile format (EMF)**
- **Graphics Interchange Format (GIF)**
- **Joint Photographic Experts Group (JPEG)**
- **Portable Network Graphics (PNG)**
- **TIFF**

For these formats, the Rel_Type child element of the ForeignData_Type element MUST specify an Image part that contains the embedded image.

Other embedded image formats and **embedded objects** are supported using fallback images.

2.2.6.1 Fallback Image

If an Image Part (section 2.3.3.5) that is in a format that is not supported (section 2.2.6) has a relationship to another Image part that is in a supported format, the latter Image part is called a fallback image and is rendered in place of the former. Otherwise, neither Image part is rendered.

2.2.7 Format

A format is a collection of properties that affect the visual appearance of shapes in a web drawing.

2.2.7.1 Fill Properties

A shape, master, or style in a web drawing can possess a variety of properties relating to the visual appearance of fills in closed geometry paths. A collection of properties defining the visual appearance of a shape, master, or style's fill is called a fill property. Each shape, master, or style has one fill property.

Fill properties allow a shape, master, or style to take on a variety of fill styles, including full transparency, solid colors, gradients, and patterns. These properties can be combined with a line property and an effect property.

The fill properties of shapes in a web drawing are specified in the Page XML Part. Each fill property is specified in a ShapeSheet_Type child element of the Shapes_Type descendant element of the PageContents element in a part.

The fill properties of masters in a web drawing are specified in the Master XML Part. Each fill property is specified in a ShapeSheet_Type child element of the Shapes_Type descendant element of the MasterContents element in a part.

The fill properties of styles in a web drawing are specified in the Document XML Part. Each fill property is specified in a StyleSheet_Type child element of the StyleSheets_Type child element of the VisioDocument element in a part.

Fill property information in shapes, masters, and styles is specified by the FillForegnd, FillForegndTrans, FillBkgnd, FillBkgndTrans, FillPattern, FillGradientDir, FillGradientAngle, FillGradientEnabled, RotateGradientWithShape, and UseGroupGradientCell_Type elements, and the Cell_Type elements belonging to the FillGradient Section_Type.

2.2.7.2 Line Properties

A shape, master, or style in a web drawing can possess a variety of properties relating to the visual appearance of lines. A collection of properties defining the visual appearance of a shape, master, or style's line is called a line property. Each shape, master, or style has one line property.

Line properties allow a shape, master, or style to take on a variety of line styles, including full transparency, solid colors, gradients, and strokes. These properties can be combined with a fill property and an effect property.

The line properties of shapes in a web drawing are specified in the Page XML Part. Each line property is specified in a ShapeSheet_Type child element of the Shapes_Type descendant element of the PageContents element in a part.

The line properties of masters in a web drawing are specified in the Master XML Part. Each line property is specified in a ShapeSheet_Type child element of the Shapes_Type descendant element of the MasterContents element in a part.

The line properties of styles in a web drawing are specified in the Document XML Part. Each line property is specified in a StyleSheet_Type child element of the StyleSheets_Type child element of the VisioDocument element in a part.

Line property information in shapes, masters, and styles is specified by the LineColor, LinePattern, LineWeight, LineCap, BeginArrow, EndArrow, LineColorTrans, CompoundType, BeginArrowSize, EndArrowSize, Rounding, LineGradientDir, LineGradientAngle, and LineGradientEnabled Cell_Type elements, and the Cell_Type elements belonging to the LineGradient Section_Type.

2.2.7.3 Effect Properties

A shape, master, or style in a web drawing can possess a variety of properties relating to effects which can affect the visual appearance of the web drawing. Each distinct effect is called an effect set. A collection of properties defining the effect sets of a shape, master, or style is called an effect property. Each shape, master, or style has one effect property consisting of distinct effect sets.

Effect properties allow a shape, master, or style to take on a variety of distinct effect sets, including shadows, bevels, glows, reflections, soft edges, sketch, and 3D rotation. These properties can be combined with a fill property and a line property.

The effect properties of shapes in a web drawing are specified in the Page XML Part. Each effect property is specified in a ShapeSheet_Type child element of the Shapes_Type descendant element of the PageContents element in a part.

The effect properties of masters in a web drawing are specified in the Master XML Part. Each effect property is specified in a ShapeSheet_Type child element of the Shapes_Type descendant element of the MasterContents element in a part.

The effect properties of styles in a web drawing are specified in the Document XML Part. Each effect property is specified in a StyleSheet_Type child element of the StyleSheets_Type child element of the VisioDocument element in a part.

2.2.7.3.1 Shadow Effect Set

A shadow effect set allows a shape, master, or style to take on one of a variety of shadows as cast by light sources of different orientations and brightness. It can be combined with other distinct effect sets. Each shape, master, or style has at most one shadow effect set.

Shadow effect set information in shapes, masters, and styles is specified by the ShdwForegnd, ShdwForegndTrans, ShdwPattern, ShapeShdwType, ShapeShdwOffsetX, ShapeShdwOffsetY, ShapeShdwObliqueAngle, ShapeShdwScaleFactor, and ShapeShdwBlur Cell_Type elements.

Shadow effect set information is partially specified in the Cell_Type elements ShdwType, ShdwOffsetX, ShdwOffsetY, ShdwObliqueAngle, and ShdwScaleFactor, in page sheets.

2.2.7.3.1.1 Shadow Distance

The Euclidean distance between the point specified by the x and y coordinates of a shadow effect set and its origin is called a shadow distance.

The x-coordinate of a shadow effect set applied to a shape, master, or style is specified by the ShapeShdwOffsetX Cell_Type element. The y-coordinate of a shadow effect set applied to a shape, master, or style is specified by the ShapeShdwOffsetY Cell_Type element.

The x-coordinate of a shadow effect set applied to a page sheet is specified by the ShdwOffsetX Cell_Type element. The y-coordinate of a shadow effect set applied to a page sheet is specified by the ShdwOffsetY Cell_Type element.

2.2.7.3.1.2 Page Default Shadow

The parts of a shadow effect set specified in Cell_Type elements in a page sheet are called a page default shadow. A shadow effect set of a shape, master, or style can be partially specified by the page default shadow specified in the page sheet of the drawing page that the shape, master, or style resides on.

If the value of the structure of a ShapeShdwType Cell_Type element in a shape, master, or style is zero, the values of the structures of the ShapeShdwType, ShapeShdwOffsetX, ShapeShdwOffsetY, ShapeShdwObliqueAngle, and ShapeShdwScaleFactor Cell_Type elements of the shape, master, or style are specified by the page default shadow during rendering. These elements are not modified.

The values of the structures of Cell_Type elements of a page sheet relating to a page default shadow specify the values of the structures of Cell_Type elements of a shape, master, or style relating to a shadow effect set according to the following table:

Page sheet Cell_Type elements	Shape, master, or style Cell_Type elements
ShdwType	ShapeShdwType
ShdwOffsetX	ShapeShdwOffsetX
ShdwOffsetY	ShapeShdwOffsetY
ShdwObliqueAngle	ShapeShdwObliqueAngle
ShdwScaleFactor	ShapeShdwScaleFactor

2.2.7.3.2 Bevel Effect Set

A bevel effect set allows a shape, master, or style to take on three-dimensional sloping edges of various types on its top and bottom faces. It can be combined with other distinct effect sets. Each shape, master, or style has at most one bevel effect set.

Bevel effect set information in shapes, masters, and styles is specified by the BevelTopType, BevelTopWidth, BevelTopHeight, BevelBottomType, BevelBottomWidth, BevelBottomHeight, BevelDepthColor, BevelDepthSize, BevelContourColor, BevelContourSize, BevelMaterialType, BevelLightingType, and BevelLightingAngle Cell_Type elements.

2.2.7.3.3 Glow Effect Set

A glow effect set allows a shape, master, or style to take on a colored, blurred outline surrounding the outer edges of the shape, master, or style. It can be combined with other distinct effect sets. Each shape, master, or style has at most one glow effect set.

Glow effect set information in shapes, masters, and styles is specified by the GlowColor, GlowColorTrans, and GlowSize Cell_Type elements.

2.2.7.3.4 Reflection Effect Set

A reflection effect set allows a shape, master, or style to take on a duplicate image of its self, reflected across its bottom edge. Transparency and blur can be applied to the duplicate image to convey the reflective properties of various surfaces. A reflection effect set can be combined with other distinct effect sets. Each shape, master, or style has at most one reflection effect set.

Reflection effect set information in shapes, masters, and styles is specified by the ReflectionSize, ReflectionTrans, ReflectionDist, and ReflectionBlur Cell_Type elements.

2.2.7.3.5 Soft Edges Effect Set

A soft edges effect set allows a shape, master, or style to take on a blur affecting its outer edges. It can be combined with other distinct effect sets. Each shape, master, or style has at most one soft edges effect set.

Soft edges effect set information in shapes, masters, and styles is specified by the SoftEdgesSize Cell_Type element.

2.2.7.3.6 Sketch Effect Set

A sketch effect set allows a shape, master, or style to take on a less polished appearance as if drawn by hand. It cannot be combined with other distinct effect sets. Each shape, master, or style has at most one sketch effect set.

Sketch effect set information in shapes, masters, and styles is specified by the SketchEnabled, SketchSeed, SketchAmount, SketchLineWeight, SketchLineChange, and SketchFillChange Cell_Type elements.

A sketch effect set renders a new geometry path for a shape, master, or style's fill and distorts both the shape, master, or style's geometry path and the fill's geometry by rendering each path segment with randomized perturbations. The geometry section of the shape, master, or style is not modified.

The value of the structure of a SketchSeed Cell_Type element is used to randomize path segment perturbations in both the geometry path and the fill's geometry. If the value of the structure of a SketchSeed Cell_Type element is equivalent for shapes, masters, or styles with identical geometry paths, the shapes, masters, or styles render identical sketch effect sets.

If a sketch effect set is active on a shape, master, or style, other effect sets do not render.

2.2.7.3.7 3D Rotation Effect Set

A 3D rotation effect set allows a shape, master, or style to take on rotations in the z-axis and perspective rotations. A 3D rotation effect set can be combined with other distinct effect sets. Each shape, master, or style has at most one 3D rotation effect set.

3D rotation effect set information in shapes, masters, and styles is specified by the RotationXAngle, RotationYAngle, RotationZAngle, RotationType, Perspective, DistanceFromGround, and KeepTextFlat Cell_Type elements.

2.2.7.4 Dynamic Theme

A shape, master, or style in a Web drawing can specify pre-defined, dynamic sets of properties which can affect its visual appearance. A set of pre-defined, dynamic properties specified in this manner is called a dynamic theme.

A dynamic theme defines properties that specify properties for color, font, fill, line properties, and effect. The properties of a dynamic theme are separated into five distinct groupings called dynamic theme components. A unique set of properties is specified by the combination of the five dynamic theme components. A shape, master, or style specifies distinct dynamic theme components from one or more dynamic themes.

The specified dynamic theme components of a shape, master, or style define more properties than the shape, master, or style can visually express at any one time. A shape, master, or style further specifies subsets of properties which actively affect its visual appearance from its specified dynamic theme components. These subsets of properties are called a quick style.

A quick style defines seven distinct subsets of properties from a shape, master, or style's specified dynamic theme components. A subset is called a quick style slice. The combination of the specified quick style slices and the specified dynamic theme components directly determines the visual appearance of the shape, master, or style.

A dynamic theme defines four distinct sets of pre-defined properties used to indirectly specify the values of properties in quick style slices in a shape, master, or style in a Web drawing. A set is called a dynamic theme variant.

2.2.7.4.1 Dynamic Theme Components

A dynamic theme defines properties that specify color, font, fill, line, and effect. It is composed of multiple parts as specified in [ISO/IEC29500-1:2016] section 20.1.6.9 and this specification.

The properties of a dynamic theme are grouped into five distinct dynamic theme components that are specified in the following table.

Dynamic theme component	Description	Location
Color scheme	Specifies a set of twelve color properties, as specified in [ISO/IEC29500-1:2016] section 20.1.6.2, and one additional color property extension, as specified in [ISO/IEC29500-1:2016] section 18.2.10.	Specified by a **clrScheme** child element as specified by the **CT_ColorScheme** type (specified in [ISO/IEC29500-1:2016] section 20.1.6.2) of a **themeElements** child element as specified by the **CT_BaseStyles** type (specified in [ISO/IEC29500-1:2016] section 20.1.6.10) of a CT_OfficeStyleSheet element in a Theme_XML_Part.
Font scheme	Specifies a set of six font properties, as specified in [ISO/IEC29500-1:2016] section 20.1.4.1.18.	Specified by a **fontScheme** child element as specified by the **CT_FontScheme** type (specified in [ISO/IEC29500-1:2016] section 20.1.4.1.18) of a **themeElements** child element as specified by the **CT_BaseStyles** type (specified in [ISO/IEC29500-1:2016] section 20.1.6.10) of a CT_OfficeStyleSheet element in a Theme_XML_Part.
Effect scheme	Specifies a set of six quick style slices of fill, line, and effect properties, as specified in [ISO/IEC29500-1:2016] section 20.1.4.1.14. These formats are used in	Specified by an **fmtScheme** child element as specified by the **CT_StyleMatrix** type (specified in [ISO/IEC29500-1:2016] section 20.1.4.1.14) of a **themeElements** child

Dynamic theme component	Description	Location
	non-connector shapes, masters, and styles.	element as specified by the **CT_BaseStyles** type (specified in [ISO/IEC29500-1:2016] section 20.1.6.10) of a CT_OfficeStyleSheet element in a Theme_XML_Part. Additional line properties and sketch effect set information are specified by CT_LineStyle child elements of a CT_SchemeLineStyles child element of a CT_LineStyles child element of an **ext** child element as specified by the **CT_OfficeArtExtension** type (specified in [ISO/IEC29500-1:2016] section 20.1.2.2.14) of an **extLst** child element as specified by the **CT_OfficeArtExtensionList** type (specified in [ISO/IEC29500-1:2016] section 20.1.2.2.15) of a **themeElements** child element as specified by the **CT_BaseStyles** type (specified in [ISO/IEC29500-1:2016] section 20.1.6.10) of a CT_OfficeStyleSheet element in a Theme_XML_Part. Additional font information is specified by CT_FontProps child elements of a CT_FontStyles child element of a CT_FontStylesGroup child element of an **ext** child element as specified by the **CT_OfficeArtExtension** type (specified in [ISO/IEC29500-1:2016] section 20.1.2.2.14) of an **extLst** child element as specified by the **CT_OfficeArtExtensionList** type (specified in [ISO/IEC29500-1:2016] section 20.1.2.2.15) of a **themeElements** child element as specified by the **CT_BaseStyles** type (specified in [ISO/IEC29500-1:2016] section 20.1.6.10) of a CT_OfficeStyleSheet element in a Theme_XML_Part.
Connector scheme	Specifies a set of six quick style slices of fill, line, and effect properties, as specified in [ISO/IEC29500-1:2016] section 20.1.4.1.14. These formats are used in connector shapes, masters, and styles.	Specified by an **fmtScheme** child element as specified by the **CT_StyleMatrix type** (specified in [ISO/IEC29500-1:2016] section 20.1.4.1.14) of a **themeElements** child element as specified by the **CT_BaseStyles** type (specified in [ISO/IEC29500-1:2016] section 20.1.6.10) of a CT_OfficeStyleSheet element in a Theme_XML_Part. Additional line properties and sketch effect set information are specified by CT_LineStyle child elements of a CT_SchemeLineStyles child element of a CT_LineStyles child element of an **ext** child element as specified by the **CT_OfficeArtExtension** type (specified in [ISO/IEC29500-1:2016] section 20.1.2.2.14) of an **extLst** child element as specified by the **CT_OfficeArtExtensionList** type (specified in [ISO/IEC29500-1:2016] section 20.1.2.2.15) of a **themeElements** child element as specified by the

Dynamic theme component	Description	Location
		CT_BaseStyles type (specified in [ISO/IEC29500-1:2016] section 20.1.6.10) of a CT_OfficeStyleSheet element in a Theme_XML_Part. Additional font information is specified by CT_FontProps child elements of a CT_FontStyles child element of a CT_FontStylesGroup child element of an **ext** child element as specified by the **CT_OfficeArtExtension** type (specified in [ISO/IEC29500-1:2016] section 20.1.2.2.14) of an **extLst** child element as specified by the **CT_OfficeArtExtensionList** type (specified in [ISO/IEC29500-1:2016] section 20.1.2.2.15) of a **themeElements** child element as specified by the **CT_BaseStyles** type (specified in [ISO/IEC29500-1:2016] section 20.1.6.10) of a CT_OfficeStyleSheet element in a Theme_XML_Part.
Primary scheme	Used in formula evaluation only.	Specified by a CT_ThemeScheme child element of a CT_LineStyles child element of an **ext** child element as specified by the **CT_OfficeArtExtension** type (specified in [ISO/IEC29500-1:2016] section 20.1.2.2.14) of an extLst child element as specified by the **CT_OfficeArtExtensionList** type (specified in [ISO/IEC29500-1:2016] section 20.1.2.2.15) of a **themeElements** child element as specified by the **CT_BaseStyles** type (specified in [ISO/IEC29500-1:2016] section 20.1.6.10) of a CT_OfficeStyleSheet element in a Theme_XML_Part.

The additional complex types in the following table that are not specified in [ISO/IEC29500-1:2016] partially specify a dynamic theme.

The additional complex types that partially specify a dynamic theme and are not specified in [ISO/IEC29500-1:2016] are listed in the following table.

Complex Type	Description
CT_LineEx	Specifies line properties information in an effect or connector scheme dynamic theme component.
CT_Sketch	Specifies sketch effect set information in an effect or connector scheme dynamic theme component.
CT_SchemeID	Specifies the index of a color, font, effect, connector or primary scheme dynamic theme component, or the GUID of a custom dynamic theme color scheme dynamic theme component.
CT_LineStyle	Specifies line properties and sketch effect set information in an effect or connector scheme dynamic theme component.
CT_LineStyles	Specifies a set of line properties and sketch effect set information in an effect

Complex Type	Description
	and connector scheme dynamic theme component.
CT_ThemeScheme	Specifies the primary scheme dynamic theme component.
CT_FmtSchemeEx	Specifies the index of an effect scheme dynamic theme component or a connector scheme dynamic theme component.
CT_SchemeLineStyles	Specifies a set of line properties and sketch effect set information in an effect or connector scheme dynamic theme component.
CT_FontProps	Specifies properties used to format a **text run**.
CT_FontStyles	Specifies a set of properties used to format a text run.
CT_FontStylesGroup	Specifies the properties used to format a text run in shapes.
CT_VarClrScheme	Specifies a color scheme list of a dynamic theme variant.
CT_VariationClrSchemeLst	Specifies four distinct color scheme lists of four distinct dynamic theme variants in a dynamic theme.
CT_VariationStyle	Specifies a style property of a style scheme list of a dynamic theme variant.
CT_VariationStyleScheme	Specifies a style scheme list of a dynamic theme variant.
CT_VariationStyleSchemeLst	Specifies four distinct style scheme lists of four distinct dynamic theme variants in a dynamic theme.

2.2.7.4.2 Dynamic Theme Identification

A shape, master, or style in a web drawing can specify distinct dynamic theme components.

The dynamic theme components used in a shape are specified in the Page XML part. Each dynamic theme component is specified by a Cell Type child element of a ShapeSheet Type child element of a Shapes Type descendant element of the PageContents element in a part.

The dynamic theme components used in a master are specified in the Master XML part. Each dynamic theme component is specified by a Cell_Type child element of a ShapeSheet_Type child element of a Shapes_Type descendant element of the MasterContents element in a part.

The dynamic theme components used in a style are specified in the Document XML part. Each dynamic theme component is specified by a Cell_Type child element of a StyleSheet Type child element of a StyleSheets Type child element of the VisioDocument element in a part.

The location of a dynamic theme component in a shape, master, or style is specified in the following table.

Dynamic theme components	Location
Color scheme	For a shape or master, the color scheme is specified by a ColorSchemeIndex Cell_Type child element of a ShapeSheet_Type element. For a style, specified by a ColorSchemeIndex Cell_Type child element of a StyleSheet_Type element.
Font scheme	For a shape or master, the color scheme is specified by a FontSchemeIndex Cell_Type

Dynamic theme components	Location
	 child element of a ShapeSheet_Type element. For a style, specified by a FontSchemeIndex Cell_Type child element of a StyleSheet_Type element.
Effect scheme	For a shape or master, the effect scheme is specified by an EffectSchemeIndex Cell_Type child element of a ShapeSheet_Type element. For a style, the effect scheme is specified by an EffectSchemeIndex Cell_Type child element of a StyleSheet_Type element.
Connector scheme	For a shape or master, the connector scheme is specified by a ConnectorSchemeIndex Cell_Type child element of a ShapeSheet_Type element. For a style, the connector scheme is specified by a ConnectorSchemeIndex Cell_Type child element of a StyleSheet_Type element.
Primary scheme	For a shape or master, the primary scheme is specified by a ThemeIndex Cell_Type child element of a ShapeSheet_Type element. For a style, the primary scheme is specified by a ThemeIndex Cell_Type child element of a StyleSheet_Type element.

2.2.7.4.3 Quick Style Slices

Quick style slices define properties that specify color, font, fill, line, and effect properties that directly affect the visual appearance of a shape, master, or style. These properties are subsets of the properties provided by the dynamic theme components specified by the shape, master, or style, and are grouped into the seven distinct quick style slices specified in the following table.

Quick style slice	Description
Line matrix	Specifies one of the six quick style slices of line properties from the effect scheme dynamic theme component for non-connector shapes, masters or styles, or from the connector scheme dynamic theme component for connector shapes, masters or styles.
Fill matrix	Specifies one of the six quick style slices of fill properties from the effect scheme dynamic theme component for non-connector shapes, masters or styles, or from the connector scheme dynamic theme component for connector shapes, masters or styles.
Effect matrix	Specifies one of the six quick style slices of effect properties from the effect scheme dynamic theme component for non-connector shapes, masters or styles, or from the connector scheme dynamic theme component for connector shapes, masters or styles.

Quick style slice	Description
Font matrix	Specifies one of six quick style slices of fonts for the font scheme dynamic theme component for shapes, masters, or styles.
Line color	Specifies one of nine colors from the color scheme dynamic theme component.
Fill color	Specifies one of nine colors from the color scheme dynamic theme component.
Shadow color	Specifies one of nine colors from the color scheme dynamic theme component.
Font color	Specifies one of nine colors from the color scheme dynamic theme component.

2.2.7.4.4 Quick Style Identification

A shape, master, or style in a web drawing can specify distinct quick style slices.

The quick style slices of a shape are specified in the Page XML part. Each quick style slice is specified by a Cell Type child element of a ShapeSheet Type child element of a Shapes Type descendant element of the PageContents element in a part.

The quick style slices of a master are specified in the Master XML part. Each quick style slice is specified by a Cell_Type child element of a ShapeSheet_Type child element of a Shapes_Type descendant element of the MasterContents element in a part.

The quick style slices of a style are specified in the Document XML part. Each quick style slice is specified by a Cell_Type child element of a StyleSheet Type child element of a StyleSheets Type child element of the VisioDocument element in a part.

The location of a quick style slice in a shape, master, or style is specified in the following table.

Quick style slices	Location
Line matrix	For a shape or master, the line matrix is specified by a QuickStyleLineMatrix Cell_Type child element of a ShapeSheet_Type element. For a style, the line matrix is specified by a QuickStyleLineMatrix Cell_Type child element of a StyleSheet_Type element.
Fill matrix	For a shape or master, the fill matrix is specified by a QuickStyleFillMatrix Cell_Type child element of a ShapeSheet_Type element. For a style, the fill matrix is specified by a QuickStyleFillMatrix Cell_Type child element of a StyleSheet_Type element.
Effect matrix	For a shape or master, the effect matrix is specified by a QuickStyleEffectsMatrix Cell_Type child element of a ShapeSheet_Type element. For a style, the effect matrix is specified by a QuickStyleEffectsMatrix Cell_Type child element of a StyleSheet_Type element.

Quick style slices	Location
Font matrix	For a shape or master, the font matrix is specified by a QuickStyleFontMatrix Cell_Type child element of a ShapeSheet_Type element. For a style, the font matrix is specified by a QuickStyleFontMatrix Cell_Type child element of a StyleSheet_Type element.
Line color	For a shape or master, the line color is specified by a QuickStyleLineColor Cell_Type child element of a ShapeSheet_Type element. For a style, the line color is specified by a QuickStyleLineColor Cell_Type child element of a StyleSheet_Type element.
Fill color	For a shape or master, the fill color is specified by a QuickStyleFillColor Cell_Type child element of a ShapeSheet_Type element. For a style, the fill color is specified by a QuickStyleFillColor Cell_Type child element of a StyleSheet_Type element.
Shadow color	For a shape or master, the shadow color is specified by a QuickStyleShadowColor Cell_Type child element of a ShapeSheet_Type element. For a style, the shadow color is specified by a QuickStyleShadowColor Cell_Type child element of a StyleSheet_Type element.
Font color	For a shape or master, the font color is specified by a QuickStyleFontColor Cell_Type child element of a ShapeSheet_Type element. For a style, the font color is specified by a QuickStyleFontColor Cell_Type child element of a StyleSheet_Type element.

A QuickStyleType Cell_Type element of a shape, master, or style specifies whether the QuickStyleLineMatrix, QuickStyleFillMatrix, and QuickStyleEffectsMatrix Cell_Type elements of the shape, master, or style refer to the effect or connector scheme dynamic theme component regardless of whether the shape, master, or style is a connector.

2.2.7.4.5 Dynamic Theme Variants

A dynamic theme (section 2.2.7.4) variant defines properties used to indirectly specify the values of properties in quick style slices.

A dynamic theme variant defines properties used to indirectly specify the value of the structure of the QuickStyleLineMatrix, QuickStyleFillMatrix, QuickStyleEffectsMatrix, QuickStyleLineColor, QuickStyleFillColor, QuickStyleShadowColor, QuickStyleFontColor, and QuickStyleFontMatrix Cell_Type elements of a shape, master, or style in a Web drawing. A dynamic theme variant also specifies embellishment and multiformat information.

The properties of a dynamic theme variant are specified in the following table.

Dynamic theme variant property	Description	Location
Color scheme list	Specifies a set of seven color properties used to indirectly specify the value of the structure of the QuickStyleLineColor, QuickStyleFillColor, QuickStyleShadowColor, and QuickStyleFontColor Cell_Type elements of a shape, master, or style in a Web drawing. Specifies multiformat information.	Specified by a CT_VarClrScheme child element of a CT_VariationClrSchemeLst child element of an **ext** child element as specified by the **CT_OfficeArtExtension** type (specified in [ISO/IEC29500-1:2016] section 20.1.2.2.14) of an **extLst** child element as specified by the **CT_OfficeArtExtensionList** type (specified in [ISO/IEC29500-1:2016] section 20.1.2.2.15) of a **CT_ColorScheme** type (specified in [ISO/IEC29500-1:2016] section 20.1.6.2) of a **themeElements** child element as specified by the **CT_BaseStyles** type (specified in [ISO/IEC29500-1:2016] section 20.1.6.10) of a CT_OfficeStyleSheet element in a Theme_XML_Part. Each color property is specified by a **srgbClr** child element as specified by the **CT_SRGBClr** type (specified in [ISO/IEC29500-1:2016] section 20.1.2.3.32) of a **CT_Color** type specified in [ISO/IEC29500-1:2016] section A.2 child element of a CT_VarClrScheme element.
Style scheme list	Specifies a set of four style properties used to indirectly specify the value of the structure of the QuickStyleLineMatrix, QuickStyleFillMatrix, QuickStyleEffectsMatrix, and QuickStyleFontMatrix Cell_Type elements of a shape, master, or style in a Web drawing. Specifies embellishment information.	Specified by a CT_VariationStyleScheme child element of a CT_VariationStyleSchemeLst child element of an **ext** child element as specified by the **CT_OfficeArtExtension** type (specified in [ISO/IEC29500-1:2016] section 20.1.2.2.14) of an **extLst** child element as specified by the **CT_OfficeArtExtensionList** type (specified in [ISO/IEC29500-1:2016] section 20.1.2.2.15) of a **themeElements** child element as specified by the **CT_BaseStyles** type (specified in [ISO/IEC29500-1:2016] section 20.1.6.10) of a CT_OfficeStyleSheet element in a Theme_XML_Part. Each style property is specified by a CT_VariationStyle child element of a CT_VariationStyleScheme type.

2.2.7.4.6 Dynamic Theme Variants Identification

A shape, master, or style in a web drawing can specify distinct dynamic theme variants.

The dynamic theme variants used in a shape are specified in the Page XML part. Each dynamic theme variant is specified by a Cell Type child element of a ShapeSheet Type child element of a Shapes Type descendant element of the PageContents element in a part.

The dynamic theme variants used in a master are specified in the Master XML part. Each dynamic theme variant is specified by a Cell_Type child element of a ShapeSheet_Type child element of a Shapes_Type descendant element of the MasterContents element in a part.

The dynamic theme variants used in a style are specified in the Document XML part. Each dynamic theme variant is specified by a Cell_Type child element of a StyleSheet_Type child element of a StyleSheets_Type child element of the VisioDocument element in a part.

The location of the properties of a dynamic theme variant in a shape, master, or style is specified in the following table.

Dynamic theme variant property	Location
Color scheme list	For a shape or master, the color scheme list is specified by a VariationColorIndex Cell_Type child element of a ShapeSheet_Type element. For a style, specified by a VariationColorIndex Cell_Type child element of a StyleSheet_Type element.
Style scheme list	For a shape or master, the style scheme list is specified by a VariationStyleIndex Cell_Type child element of a ShapeSheet_Type element. For a style, specified by a VariationStyleIndex Cell_Type child element of a StyleSheet_Type element.

2.2.7.4.7 Dynamic Theme Functions

The properties specified by a dynamic theme of a shape (section 2.2.3), master, or style are referenced through two function tokens persisted in formula expressions in a web drawing.

The ThemeVal function token, when called without argument, returns the property value from the dynamic theme for the Cell_Type child element that it resides in directly without invoking theme inheritance. The ThemeVal function token, when called with an argument, returns the property value from the dynamic theme specified by the argument directly without invoking theme inheritance.

The ThemeProp function token accepts an argument to retrieve the multiformat and embellishment property values from a dynamic theme (section 2.2.7.4) as specified by the argument.

2.2.7.4.8 Custom Dynamic Theme Color Scheme

The set of color properties in a dynamic theme can be specified by a master instead of a color scheme dynamic theme component. A set of color properties specified in this manner is called a custom dynamic theme color scheme.

The set of color property values in a custom dynamic theme color scheme is specified by the Value Cell_Type child elements of the msvThemeDarkColor, msvThemeLightColor, msvThemeAccentColor, msvThemeAccentColor2, msvThemeAccentColor3, msvThemeAccentColor4, msvThemeAccentColor5, msvThemeAccentColor6 and msvThemeBackgroundColor Row_Type child elements of a User Section_Type element descendant of a MasterContents element of a master.

A custom dynamic theme color scheme is specified by a CT_SchemeID child element of a **ext** child element as specified by the **CT_OfficeArtExtension** type (specified in [ISO/IEC29500-1:2016] section 20.1.2.2.14) of an **extLst** child element as specified by the **CT_OfficeArtExtensionList** type (specified in [ISO/IEC29500-1:2016] section 20.1.2.2.15) of a **themeElements** element as specified by the **CT_BaseStyles** type (specified in [ISO/IEC29500-1:2016] section 20.1.6.10) in a Theme_XML_part.

If the value of the **schemeEnum** attribute of the CT_SchemeID element is equal to 65535 and the value of the **schemeGUID** attribute of the CT_SchemeID element is equal to the value of the **UniqueID** attribute of the Master_Type child element of a Masters_Type element in the Masters XML part, the custom dynamic theme color scheme is specified by the master with the matching **UniqueID** attribute.

2.2.7.4.9 Connector

A shape, master, or style in a web drawing can be either a connector or a non-connector in terms of a dynamic theme.

If a shape, master, or style inherits from a style sheet whose **NameU** attribute value is equal to "Connector", the shape, master, or style is a connector; otherwise, the shape, master, or style is a non-connector.

2.2.7.4.10 Embellishment and Multiformat

A dynamic theme variant of a dynamic theme specifies two PtgByte parse tokens that are used in formula evaluation only. One is called **embellishment**, and the other is called **multiformat**.

If the value of the structure of the EmbellishmentIndex Cell_Type element of a shape is equal to 0, embellishment is specified by the value of the **embellishment** attribute of a CT_VariationStyleScheme child element, specified by the VariationStyleIndex Cell_Type element of a shape, of a CT_VariationClrSchemeLst child element of an **ext** child element as specified by the **CT_OfficeArtExtension** type (specified in [ISO/IEC29500-1:2016] section 20.1.2.2.14) of an **extLst** child element as specified by the **CT_OfficeArtExtensionList** type (specified in [ISO/IEC29500-1:2016] section 20.1.2.2.15) of a **themeElements** child element as specified by the **CT_BaseStyles** type (specified in [ISO/IEC29500-1:2016] section 20.1.6.10) of a CT_OfficeStyleSheet element in a Theme_XML_Part. Otherwise, embellishment is specified by the value of the structure of the EmbellishmentIndex Cell_Type element.

Multiformat is specified by the value of the **monotone** attribute of a CT_VariationClrScheme child element, specified by the VariationColorIndex Cell_Type element of a shape, of a CT_VariationClrSchemeLst child element of an **ext** child element as specified by the **CT_OfficeArtExtension** type (specified in [ISO/IEC29500-1:2016] section 20.1.2.2.14) of an **extLst** child element as specified by the **CT_OfficeArtExtensionList** type (specified in [ISO/IEC29500-1:2016] section 20.1.2.2.15) of a **CT_ColorScheme** type (specified in [ISO/IEC29500-1:2016] section 20.1.6.2) of a **themeElements** child element as specified by the **CT_BaseStyles** type (specified in [ISO/IEC29500-1:2016] section 20.1.6.10) of a CT_OfficeStyleSheet element in a Theme_XML_Part.

2.2.7.5 Fixed Theme

A shape, master, or style in a web drawing can specify pre-defined, fixed sets of properties which can affect its visual appearance. A set of pre-defined, fixed properties specified in this manner is called a fixed theme.

A fixed theme defines properties that specify color, font, fill, line, and effect properties. The properties of a fixed theme are separated into two groupings: a fixed color scheme and a fixed effect scheme. A shape, master, or style specifies a fixed theme by specifying a fixed color scheme and a fixed effect scheme.

The set of property values in a fixed color scheme is specified by the vThemeColor custom structure. The set of property values in a fixed effect scheme is specified by the vThemeEffect custom structure.

A fixed color scheme is specified by the Value Cell_Type child element of an msvThemeColors Row_Type child element of a User Section_Type element in a shape, master, or style. A fixed effect

scheme is specified by the Value Cell_Type child element of an msvThemeEffects Row_Type child element of a User Section_Type element in a shape, master, or style.

A Theme function token returns the property values from the set of properties specified by a fixed color scheme and a fixed effect scheme of a shape, master, or style.

2.2.7.5.1 Custom Fixed Color and Effect Schemes

The set of property values specified by a fixed color scheme or a fixed effect scheme can be specified by a master instead of a vThemeColor or vThemeEffect custom structure. A fixed color scheme of this description is called a custom fixed color scheme. A fixed effect scheme of this description is called a custom fixed effect scheme. A shape, master, or style in a web drawing can specify a custom fixed color scheme in place of a fixed color scheme and/or a custom fixed effect scheme in place of a fixed effect scheme.

The set of property values in a custom fixed color scheme is specified by the Value Cell_Type child elements of the msvThemeTextColor, msvThemeFillColor, msvThemeFillColor2, msvThemeLineColor, msvThemeConnectorColor, msvThemeShadowColor, msvThemeAccentColor, msvThemeAccentColor2, msvThemeAccentColor3, msvThemeAccentColor4, msvThemeAccentColor5, and msvThemeBackgroundColor Row_Type child elements of a User Section_Type element in a master.

The set of property values in a custom fixed effect scheme is specified by the Value Cell_Type child elements of the msvThemeLatinFont, msvThemeAsianFont, msvThemeComplexFont, msvThemeLineTransparency, msvThemeLineWeight, msvThemeLinePattern, msvThemeLineRounding, msvThemeConnectorTransparency, msvThemeConnectorPattern, msvThemeConnectorWeight, msvThemeConnectorRounding, msvThemeConnectorBegin, msvThemeConnectorEnd, msvThemeConnectorEnd2, msvThemeConnectorBeginSize, msvThemeConnectorEndSize, msvThemeFillTransparency, msvThemeFillPattern, msvThemeShadowTransparency, msvThemeShadowPattern, msvThemeShadowStyle, msvThemeShadowXOffset, msvThemeShadowYOffset, msvThemeShadowMagnification, and msvThemeShadowDirection Row_Type child elements of a User Section_Type element in a master.

A custom fixed color scheme is specified by the Value Cell_Type child element of an msvThemeColors Row_Type child element of a User Section_Type element in a shape, master, or style. If the **V** attribute of the Value Cell_Type element is equal to 254 and the argument of the USE function token of the **F** attribute of the Value Cell_Type element is equal to the **UniqueID** attribute of the master specified by the Master_Type child element of a Masters_Type element in the Masters XML part, the custom fixed color scheme of the shape, master, or style is specified by the master.

A custom fixed effect scheme is specified by the Value Cell_Type child element of an msvThemeEffects Row_Type child element of a User Section_Type element in a shape, master, or style. If the **V** attribute of the Value Cell_Type element is equal to 254 and the argument of the USE function token of the **F** attribute of the Value Cell_Type element is equal to the **UniqueID** attribute of a master specified by the Master_Type child element of a Masters_Type element in the Masters XML Part, the custom fixed effect scheme of the shape, master, or style is specified by the master.

A Theme function token returns the property values from the set of properties specified by a custom fixed color scheme and/or a custom fixed effect scheme of a shape, master, or style.

2.2.7.6 Color Table

A color value in a web drawing can be specified as either a PtgColorRGB parse token or an unsigned long integer.

If a color value specified as an unsigned long integer is greater than or equal to zero and less than or equal to 23, the color-value of the specified color is specified by the following table.

Unsigned Long Integer	Color-value
0	#000000
1	#FFFFFF
2	#FF0000
3	#00FF00
4	#0000FF
5	#FFFF00
6	#FF00FF
7	#00FFFF
8	#800000
9	#008000
10	#000080
11	#808000
12	#800080
13	#008080
14	#C0C0C0
15	#E6E6E6
16	#CDCDCD
17	#B3B3B3
18	#9A9A9A
19	#808080
20	#666666
21	#4D4D4D
22	#333333
23	#1A1A1A

If a color value specified as an unsigned long integer is greater than 23, the **RGB** value of the specified color is specified by a ColorEntry_Type child element of a Colors_Type child element of a VisioDocument element in a Document XML part. If the value of an **IX** attribute of a ColorEntry_Type element is equal to the specified unsigned long integer, the RGB value of the specified color is equal to the **RGB** attribute specified by the ColorEntry_Type element.

2.2.7.7 Font Table

A font table specifies the **fonts** used in a web drawing. It is specified by a FaceNames_Type child element of a VisioDocument element in a Document XML part. Each font is specified by a FaceName_Type child element of the FaceNames_Type element.

A shape, master, or style specifies its fonts using the Font, AsianFont, and ComplexScriptFont Cell_Type elements. If the value of the **V** attribute of a Font, AsianFont, or ComplexFont Cell_Type element of a shape, master, or style is equal to the value of the **NameU** attribute of a FaceName_Type element, the shape, master, or style specifies the font specified by the FaceName_Type element.

2.2.7.8 Custom Pattern

A fill, line, or line end in a shape, master, or style in a web drawing can be specified by a master. A fill, line, or line end specified in this way is called a custom pattern.

Custom patterns are specified by a Masters XML part. Each custom pattern is specified in a Master_Type child element of a Masters_Type element in a part. The **PatternFlags** attribute of a Master_Type element specifies the type and behavior of the custom pattern.

The visual appearance of a custom pattern is specified by the shapes in the Master XML part that corresponds to the master. Each shape is specified by a ShapeSheet_Type child element of the Shapes_Type descendant element of the MasterContents element in a part.

A shape, master, or style in a web drawing specifies a custom pattern according to the following table.

Custom pattern	Description
Fill	Specified by a FillPattern Cell_Type element whose **V** attribute value is equal to 254. If the argument of the USE function token of the **F** attribute of the FillPattern Cell_Type element is equal to the **NameU** attribute of a master specified by a Master_Type element, the shape, master or style specifies a fill custom pattern specified by a Master_Type element.
Line	Specified by a LinePattern Cell_Type element whose **V** attribute value is equal to 254. If the argument of the USE function token of the **F** attribute of the LinePattern Cell_Type element is equal to the **NameU** attribute of a master specified by a Master_Type element, the shape, master, or style specifies a line custom pattern specified by a Master_Type element.
Line end	Specified by a BeginArrow or EndArrow Cell_Type element whose **V** attribute value is equal to 254. If the argument of the USE function token of the **F** attribute of the BeginArrow or EndArrow Cell_Type element is equal to the **NameU** attribute of a master specified by a Master_Type element, the shape, master, or style specifies a line end custom pattern specified by a Master_Type element.

2.2.7.9 Data Formatting

The text field or shape data in a shape or master in a web drawing can specify a format that affects the visual appearance of its **field**. Formatting specified in this manner is called a data format.

2.2.7.9.1 Text Field Data Formatting

The text field in a shape or master in a web drawing can specify a data format that affects the visual appearance of its **field** that is used in a **text run**.

A text field data format is specified in a Field Section_Type element in a shape or master. Each data format is specified by a Row_Type child element of the Field Section_Type element. A Value Cell_Type

child element of the Row_Type element specifies a value to be formatted. A Format Cell_Type child element of the Row_Type element specifies the data format to apply to the value specified by its sibling Value Cell_Type.

The **V** attribute of the Format Cell_Type element is a vFormatString custom structure that specifies the data format information designating how the value is displayed.

2.2.7.9.2 Shape Data Formatting

The shape data in a shape or master in a web drawing can specify a data format that affects the visual appearance of its shape data **field**.

The shape data formats are specified in a Property Section_Type element in a shape or master. The data format for a shape data field is specified by its corresponding Row_Type child element of the Property Section_Type element. A Value Cell_Type child element of the Row_Type element specifies a shape data field value to be formatted.

The **V** attribute of the Value Cell_Type element specifies the shape data field value. A Type Cell_Type child element of the Row_Type element specifies the type of shape data field value that is stored in its sibling Value Cell_Type element. The **V** attribute of the Type Cell_Type element is a vDataType custom structure that specifies the shape data field value type. A Format Cell_Type child element of the Row_Type element specifies the data format to apply to shape data field value specified by its sibling Value Cell_Type. The **V** attribute of the Format Cell_Type element is a vFormatString custom structure that specifies the data format information designating how the shape data field value is displayed.

2.2.8 Text

A shape or master can contain text that is specified by one or more **text runs**. The text runs associated with a shape are specified by the contents of a Text_Type element contained in the ShapeSheet_Type element of the shape. The characters in a text run can be specified explicitly or can be a reference to a text field.

A text run has characters and properties of character, paragraph, and tabs specified as follows:

- Character properties are specified by cp_Type elements.
- Paragraph properties are specified by pp_Type elements.
- Tabs properties are specified by tp_Type elements.
- Text fields are specified by fld_Type elements.

The content of a Text_Type element is composed of the text characters associated with the shape, interspersed with cp_Type, pp_Type, tp_Type, and fld_Type elements.

The beginning of a text run on a shape is specified by a Text_Type, cp_Type, pp_Type, or tp_Type element.

2.2.8.1 Character Properties

The cp_Type element in a shape or master specifies the beginning of a **text run** and the set of character properties used for the text run. These character properties are used until the end of the Text_Type element, or until another cp_Type element specifies new character properties.

The cp_Type element specifies the index of a Row_Type element that is contained in a Character Section_Type element. This Row_Type element specifies the information about the character properties using a collection of Cell_Type elements. It is either contained under the ShapeSheet_Type element for the shape or it is inherited.

A collection of Cell_Type elements that define the character properties are composed of AsianFont, Case, Color, ColorTrans, ComplexScriptFont, ComplexScriptSize, DblUnderline, DoubleStrikethrough, Font, FontScale, LangID, LetterSpace, Overline, Pos, Size, Strikethru, and Style Cell_Type elements.

2.2.8.2 Paragraph Properties

The pp_Type element in a shape or master specifies the beginning of a **text run** and the set of paragraph properties used for the text run. These paragraph properties are used until the end of the Text_Type element, or until another pp_Type element specifies new paragraph properties.

The pp_Type element specifies the index of a Row_Type element that is contained in a Paragraph Section_Type element. This Row_Type element specifies the information about the paragraph properties using a collection of Cell_Type elements. It is either contained under the ShapeSheet_Type element for the shape or it is inherited.

A collection of Cell_Type elements that define the paragraph properties are composed of Bullet, BulletFont, BulletFontSize, BulletStr, Flags, HorzAlign, IndFirst, IndLeft, IndRight, SpAfter, SpBefore, SpLine, and TextPosAfterBullet Cell_Type elements.

2.2.8.3 Tabs Properties

The tp_Type element in a shape or master specifies the beginning of a **text run** and the set of tab stops used for the text run. These tab stops are used until the end of the Text_Type element, or until another tp_Type element specifies new tab stops.

The tp_Type element specifies the index of a Row_Type element that is contained in a Tabs Section_Type element. This Row_Type element specifies the information about the tab stops using a collection of Cell_Type elements. It is either contained under the ShapeSheet_Type element for the shape or it is inherited.

A Row_Type element in a Tabs Section_Type element contains a series of Position and Alignment Cell_Type element pairs with **N** attributes equal to Position# and Alignment#, where the # represents the tab stop index. A Position and Alignment pair specifies the stop position and alignment for a single tab stop.

2.2.8.4 Text Fields

The fld_Type element in a shape or master specifies a **field** that is used in a **text run**. It specifies the index of a Row_Type element that is contained in a Field Section_Type element. This Row_Type element specifies the information about the field using a collection of Cell_Type elements. It is either contained under the ShapeSheet_Type element for the shape or it is inherited.

If the value of the **IX** attribute of a fld_Type element is equal to the value of the **IX** attribute of a Row_Type element that is contained in a Field Section_Type element in the shape or master, the Cell_Type elements contained under the Row_Type element specify information about the field of the fld_Type element.

A collection of Cell_Type elements that define a text field composed of Calendar, Format, ObjectKind, Type, and Value Cell_Type elements. The Value Cell_Type element specifies the value of the field. The Calendar, Format, ObjectKind, and Type Cell_Type elements specify how the value of the field is displayed in the text run.

2.2.8.5 Text Block

The **text runs** associated with a shape are rendered using a rectangular composition area called a text block. A text block specifies information related to the visual appearance of the text runs as a whole.

A text block uses a collection of Cell_Type elements contained under the ShapeSheet_Type element for a shape to specify position, transform, margin, alignment, direction, and background information for the text runs associated with the shape. A collection of Cell_Type elements that define how text is arranged in the composition area of a text block that is detailed in the following table.

Cell_Type element(s)	Description
TxtPinX, TxtPinY, TxtLocPinX, and TxtLocPinY	Specifies the text block coordinate system.
TxtAngle	Specifies the angle of counterclockwise rotation of the text block in the coordinate system of the shape it is associated with.
TxtWidth and TxtHeight	Specifies the width and height of the text block.
LeftMargin, RightMargin, TopMargin, and BottomMargin	Specifies the positioning of the text runs against the borders of the text block.
TextDirection	Specifies whether the text runs are rendered in an upright alignment with the top border of the text block or in an upright alignment with the right border of the text block within the text block coordinate system.
VerticalAlign	Specifies the vertical alignment of the text runs. If the value of the TextDirection Cell_Type element structure is equal to zero, text runs are rendered starting from the top border, middle, or bottom border of the text block within the text block coordinate system. If the value of the TextDirection Cell_Type element structure is equal to one, text runs are rendered starting from the right border, center, or left border of the text block with the text block coordinate system.
TextBkgnd	Specifies the solid fill color property of the background of the text block.
TextBkgndTrans	Specifies the transparency level of the solid fill color property of the background of the text block.

2.2.8.5.1 Text Block Coordinate System

A point on a text block is specified by coordinates on a two-dimensional Cartesian plane where the x-coordinate specifies the horizontal position and the y-coordinate specifies the vertical position. Every text block defines a coordinate system.

The TxtPinX and TxtPinY Cell_Type child elements of a ShapeSheet_Type element of a shape specify the pin of the text block in the coordinate system of the shape. The TxtLocPinX and TxtLocPinY Cell_Type child elements of a ShapeSheet_Type element of a shape specify the pin of the block in local coordinates.

A point on a text block specified in local coordinates can be converted into its associated shape coordinates by applying transformations in the following order:

1. Subtract the value of the TxtLocPinX Cell_Type element from the x-coordinate.
2. Subtract the value of the TxtLocPinY Cell_Type element from the y-coordinate.
3. Mirror the point about the y-axis if the value of the FlipX Cell_Type element is equal to one.
4. Mirror the point about the x-axis if the value of the FlipY Cell_Type element is equal to one.
5. Rotate the point counterclockwise around the origin by the value of the TxtAngle Cell_Type element.
6. Add the value of the TxtPinX Cell_Type element to the x-coordinate.
7. Add the value of the TxtPinY Cell_Type element to the y-coordinate.

2.2.9 Comments

Comments are plain text annotations in a web drawing. Each comment has an associated author and drawing page. It can have an associated shape on the drawing page. A collection of comments in a web drawing is specified by a Comments XML part.

A Comments_Type element in a Comments XML part contains the AuthorList_Type element and the CommentList_Type elements, which specifies the comment authors and comments respectively.

Each AuthorEntry_Type child of an AuthorList_Type parent element contains information for a single author. An author can be associated with one or more comments. The author is uniquely identified by the **ID** and **ResolutionID** attributes. Additional author information is provided by the **Name** and **Initials** attributes.

Each CommentEntry_Type child of a CommentList_Type parent element represents a single comment. The **text runs** associated with a comment are specified by the contents of a CommentEntry_Type element. The following attributes specify additional properties of the comment:

- The **AuthorID** attribute specifies the author of a comment. This attribute is equal to the **ID** attribute of the AuthorEntry_Type element that corresponds to the author.
- The **PageID** attribute specifies the page a comment refers to. This attribute is equal to the **ID** attribute of a Page_Type element of the drawing page.
- The **ShapeID** attribute can specify a shape on the drawing page that the comment refers to. When the **ShapeID** attribute exists, it is equal to the **ID** attribute of the ShapeSheet_Type element of the shape.

2.2.10 Data Connectivity and Refresh

This section describes how **data sources** can be referenced, queried and connected to from within a web drawing.

2.2.10.1 Data Connections

A web drawing can be linked to data in databases and other **data sources** which can affect various attributes of the web drawing, including its visual appearance. A relationship to such data sources is called a data connection.

A data connection contains properties that specify how the application connects to and queries the data source, including the type of **data provider** (for example, **OLE DB** or **ODBC**) required to access

a data source, the name of the server on which the data source is hosted, security information to access the data source, and a **query** to execute on the server.

The data connections in a web drawing are specified by the Connections XML part. Each data connection is specified by a DataConnection_Type child element of the DataConnections element in a part.

Data connection information can be specified solely by the DataConnection_Type element or partially by information in an external file. If the **FileName** attribute of the DataConnection_Type element is empty, a data connection is solely specified by the DataConnection_Type element. If the **FileName** attribute is not empty, a data connection is specified by the DataConnection_Type element and the information contained in the file found at the path described by the value of the **FileName** attribute.

The following elements in parts of a web drawing specify supplementary information about the data connection.

- A DataRecordSet_Type element in the Recordsets XML Part contains a **ConnectionID** attribute that is equal to the **ID** attribute of the DataConnection_Type element for the data connection and specifies a recordset that uses this data connection to connect to and query a data source.

Data connections can be established for the types of data sources listed in the **ConnectionString** attribute of the DataConnection_Type element.

2.2.10.2 Recordset

A recordset is the data that is returned from a **data source**, organized into sets of **rows** and **fields**. The recordset is related to a specific data source using a data connection. The operation of replacing the contents of a recordset with data queried from a data source, using the associated data connection, is called refreshing the recordset.

The rows of a recordset can be linked to shapes in drawing pages in a web drawing through data binding. This allows additional properties of the web drawing to be updated when the recordset is refreshed.

The recordsets in a web drawing are specified by the Recordsets XML part. Each recordset is specified by a DataRecordSet_Type child element of the DataRecordSets element in a part. The fields of the recordset are specified by the DataColumns_Type child element of the DataRecordSet_Type element.

2.2.10.2.1 Data Binding

Data binding is the association between a **row** of a recordset and a shape in a drawing page. A row can be bound to zero or more shapes. A shape can have zero or one recordset rows bound to it.

The rows of a recordset that are bound to shapes are specified by the RowMap_Type child elements of the DataRecordSet_Type element for the recordset. In each RowMap_Type element, the row is identified by a **RowID** attribute, the shape is identified by a **ShapeID** attribute, and the drawing page containing the shape is identified by a **PageID** attribute.

The **fields** of a recordset are mapped to shape data items in the bound shapes. A field can be mapped to zero or one shape data item in each shape that is bound to a row in the recordset. A shape data item can have zero or one fields bound to it.

The mapping between each field of the recordset and shape data item of the bound shape is specified by the DataColumn_Type element for the field and a Row_Type child element of the Property Section_Type element for the shape. A mapping exists if there is a Row_Type element with an **N** attribute that is equal to the **Name** attribute of the DataColumn_Type element.

2.2.10.3 Recordset Refresh

A recordset refresh is the operation of replacing the contents of a recordset with data queried from a **data source**. Refresh information is specified by the recordset and its associated data connection.

The refresh **query** is specified by the **Command** attribute of the DataRecordSet_Type element for the recordset. If the **Command** attribute of the DataRecordSet_Type element is empty, the refresh query is specified by the **Command** attribute of the DataConnection_Type element for the data connection.

Only recordsets that are enabled for data refresh participate in refresh operations. A recordset is enabled for data refresh when both of the following conditions are true:

- A PublishSettings_Type child element of the VisioDocument_Type element for the web drawing is missing, or the PublishSettings_Type element contains a RefreshableData_Type child element with an **ID** attribute equal to the **ID** attribute of the DataRecordSet_Type element for the recordset.
- The DataRecordSet_Type element for the recordset contains an **Options** attribute with a value that is not a bitwise OR combination of the value one.

When the data in the **rows** of a recordset change, shape data in shapes with data bindings to the recordset are also updated. The **RefreshOverwriteAll** attribute of the DataRecordSet_Type element for the recordset determines which shape data items are updated. Individual shape data items are then updated in the following manner:

- If the cell associated with the shape data item contains a formula expression containing a Guard function token, the shape data item is not updated.
- If the cell associated with the shape data item contains a formula expression containing a SetAtRef function token, the value of the cell referenced by the first argument of the function is updated with the value of the recordset for the mapped **field** and row.
- Otherwise, the value of the shape data item is updated with the value of the recordset for the mapped field and row. This could involve a data type conversion from the data type of the field, as specified by the **DataType** attribute of its corresponding DataColumn_Type element, to the data type of the shape data item.

All formulas in cells that have been updated are recalculated as part of a diagram update.

2.2.10.4 Recordset Row Addressing

Specific recordset **rows** are tracked across a recordset refresh operation using a **primary key**. A recordset can explicitly specify a primary key, or it can specify that the current ordering of the rows be used as a primary key.

If the **RowOrder** attribute of the DataRecordSet_Type element for the recordset is zero, the primary key is specified by a collection of PrimaryKey_Type child elements of the DataRecordSet_Type element. If the **RowOrder** attribute is one, the primary key is specified by the position of each row in the recordset regardless of its contents.

2.2.11 Diagram Update

This section describes how the properties of a web drawing are changed from their current state to an updated state by a diagram update operation. A diagram update is initiated following a recordset refresh or through update triggers. These actions each specify a set of properties to change.

Additional properties of the web drawing can have formulas that are dependent on the initial set of updated properties. Expressions in the formulas are evaluated to calculate new property values.

2.2.11.1 Update Triggers

An update trigger is a structure in a web drawing that signifies a diagram update is needed. The trigger is specified by the presence of a special function token in the formula expression of a property of the web drawing.

The Category, Creator, Description, Directory, DocLastEdit, DocLastSave, Keywords, Now, Subject, and Title function tokens specify update triggers.

The Trigger_Type element specifies one or more drawing pages that contain a specific update trigger. The **N** attribute of the Trigger_Type element determines the update trigger and the possible values for the **N** attribute are defined in the Triggers section of this specification.

2.2.11.2 Formulas

The properties that are specified in cells can have formulas. Formulas specify how the properties of a web drawing are modified during a diagram update operation.

A formula is specified by the **F** attribute of a Cell_Type child element in a Section_Type, Row_Type, ShapeSheet_Type, PageSheet_Type, StyleSheet_Type, or DocumentSheet_Type element.

The following sections describe the concepts and elements of a formula.

2.2.11.2.1 Formula Expression

A formula expression is a sequence of functions, values, and references that make up a formula and that produce a value when evaluated.

A formula expression contains a sequence of parse tokens. The Formula ABNF and Full Grammar Definition section in this specification defines the valid formula expressions in a web drawing.

2.2.11.2.2 Parse Tokens

A parse token is a string of characters that specifies a **token** in a formula expression. A parse token in a web drawing is a function, an operand, or a reference token.

2.2.11.2.2.1 Function Tokens

A function token represents a function in a formula expression. The Formula ABNF and Full Grammar Definition section in this specification defines the valid function tokens in a formula expression. The syntax for each function token is described in the Function Token Definitions section.

A function can specify a set of arguments used in the evaluation of the function token. The arguments of a function are additional parse tokens in the formula expression. The value returned by an evaluated function is an operand token.

2.2.11.2.2.2 Operand Tokens

An operand token represents a value in a formula expression. This token can be either the solitary value in a formula, an argument of a function, the evaluation result of a function, or the evaluation result of a cell reference.

The Formula ABNF and Full Grammar Definition section in this specification defines the valid operand tokens in a formula expression. The syntax for each operand token is described in the Parse Token Definitions section.

In addition to its use in a formula expression, an operand token also specifies a single value that can be persisted in the file and represents one of the tokens specified in the token group vAny.

An operand token can have **Value**, **Unit**, **Dimension**, **Currency**, and **Error State** properties.

The **Value** of an operand token is the value of the structure. When stored in a Cell_Type element, the **Value** of an operand token is stored in the **V** attribute with the following exceptions:

- For a Boolean value, the **Value** is "FALSE" or "TRUE" but is stored as zero or one, respectively, in the Cell_Type element **V** attribute.
- For a currency value, both the **Value** and **Currency** are stored in the **V** attribute.
- For a multi-dimensional value, the **Value**, **Unit**, and **Dimension** are stored in the **V** attribute.
- For a two-dimensional point, the **Value** and **Unit** are stored in the **V** attribute.
- For an error code, the operand token has no **Value**.

The **Unit**, **Dimension**, and **Currency** of an operand token give additional meaning to the token's **Value**. Not all operand tokens have a **Unit**, **Dimension**, or **Currency**. When stored in a Cell_Type element, the **Unit** of an operand token is stored in the **U** attribute.

A **Dimension** is not persisted unless the token is a PtgNumMultiDim. For a PtgNumMultiDim, the **Value** and the **Dimension** are stored in the **V** attribute as specified by the PtgNumMultiDim format.

Currency values are the only operand tokens to have a **Currency**. For a currency value, the **Value** is concatenated with the **Currency** and stored in the **V** attribute as specified by the PtgCy parse token format.

The **Error State** of an operand token represents an error obtained during formula evaluation. Depending on the function, the **Error State** of an operand token can either be used or ignored during formula evaluation. When stored in a Cell_Type element, the **Error State** of an operand token is stored in the **E** attribute.

An operand token represents, and can be converted into, one of the following types of values.

- A string value
- A numeric value
- A Boolean value
- A currency value
- A color value
- A date value
- A geometry function value
- An error value

These conversions translate many different source operand tokens into tokens representing different classes of inputs that are required by functions. Functions can operate on the converted tokens but can also refer to elements of the source token. See the Custom Input Types section for details on common token conversions used by functions.

2.2.11.2.2.2.1 String Values

A string value represents textual information and is specified as a PtgString parse token. For a string operand token, the **Value** property is the string and the **Unit** property is equal to "STR". The token does not have a **Dimension** or **Currency** property.

Other tokens can specify a string value according to the conversion specified in the vString custom input type.

2.2.11.2.2.2.2 Numeric Values

A numeric value represents a number with or without units. Boolean values, currency values, color values, and date values are classified separately.

A numeric value is specified as one of the tokens in the custom token group vNum (except PtgDate). Other token types can also represent numeric values as specified in the vDouble, vFloat, vSignedInt, vSignedLong, vUnsignedInt, and vUnsignedLong custom input types.

Numeric values that represent length, angle, duration, and typographic units, as well as higher dimensional forms of these units, are described in the Unit Number section. These numeric values have the special property so that their **Value** property is specified as a Custom Internal Unit Type. When found in a formula expression, the **Value** is converted to the **Unit** and **Dimension** properties of the operand token; this is called the display value. During formula evaluation, the operand token **Value** (not the display value) from the formula expression is used.

A numeric value that represents a percentage value is specified as a PtgNumPct parse token. The **Value** is a number as a fraction of 100, the **Unit** is equal to "PER", and the **Dimension** is zero.

If the numeric value has no units, it represents a number and is persisted in the file as a PtgNum parse token or equivalent member of the vScalar custom token group. The **Value** is equal to the numeric value, and the **Dimension** is zero. It does not have a **Unit**.

2.2.11.2.2.2.3 Boolean Values

A value that represents a Boolean value is specified as a PtgBool parse token. The **Value** property is either "FALSE" or "TRUE", the **Unit** is equal to "BOOL" or does not exist, and the **Dimension** property is zero. It does not have a **Currency** property.

When stored in a Cell Type element, the **Value** of an operand token is converted to zero or one, where zero represents "FALSE" and one represents "TRUE", and is stored in the **V** attribute.

Other tokens can also represent a Boolean value according to the conversion specified in the vBoolean custom input type.

2.2.11.2.2.2.4 Currency Values

A value that represents a currency is specified as a PtgCy parse token. No other token type can represent a currency value. The only custom input type that preserves both a currency value and its associated currency is vDoubleEx.

The **Value** is the numeric value of the currency, the **Currency** is the associated currency string as specified in vCurrency custom structure, the **Unit** is equal to "CY" and the **Dimension** is zero.

2.2.11.2.2.2.5 Color Values

A value that represents a **red-green-blue (RGB)** color value is specified as a PtgColorRGB parse token. The **Value** represents the hexadecimal value of the color or the index in the color table, the **Unit** property is equal to "COLOR" or does not exist, and the **Dimension** property is zero. It does not have a **Currency** property.

Other tokens can also represent a color value according to the conversion specified in the vColor custom input type.

2.2.11.2.2.2.6 Date Values

A value that represents a date is specified as a PtgDate parse token. The **Value** property is a date and time of day, the **Unit** property is equal to "DATE", and the **Dimension** property is one. It does not have a **Currency** property.

Other token types can represent a date according to the conversion specified in the DateTime function.

2.2.11.2.2.2.7 Geometry Function Values

A geometry function value represents geometry path data that is specified by a PtgPnt, PtgNURBS, or PtgPolyline parse token. The **Value** property is a set of numeric values that specify individual properties of the geometry path. These numeric values are arranged in a syntax that matches the Pnt, NURBS, and Polyline function token definitions.

The **Unit** property of the token is equal to "PNT" for a PtgPnt or "POLYLINE" for a PtgPolyline; the PtgNURBS does not have a **Unit**. The **Dimension** property is zero, and it does not have a **Currency** property. No other token types can represent a geometry function value.

2.2.11.2.2.2.8 Error Values

An error code that is returned as a result of a formula evaluation is specified as a PtgErr parse token. When a function encounters a PtgErr as one of its arguments, it returns the same error value. The exceptions are the functions IsErr, IsErrNA, IsError, and IsErrValue, which are specifically designed to detect specific error values.

2.2.11.2.2.3 Reference Tokens

A reference token represents a cell, other than the cell containing the formula expression, whose value is used in the evaluation of a formula expression. A reference token allows a formula expression to depend on the values of other properties in the web drawing.

The Formula ABNF and Full Grammar Definition section defines the valid reference tokens in a formula expression. The syntax for each reference token is described in the Reference Token Definitions section.

The result of a reference token that is evaluated is an operand token.

2.2.11.2.3 Formula Evaluation

Formula evaluation is the process of taking a complex formula expression and computing a single resulting parse token.

The parse tokens that make up the formula expression are evaluated in sequence as specified by the Order of Operations. Each function token and reference token in the formula expression is evaluated to produce an operand token.

The logic for evaluating a particular function token is specified by the Function Token Definitions. A reference token is evaluated by returning the value of the cell specified by the reference token. Functions and references are evaluated within a reference context, which is the specification of the sheet containing the properties to be used in the evaluation.

When the formula expression of a cell is evaluated, the formula expressions of other cells that contain reference tokens that reference the cell are also evaluated.

2.2.11.2.4 Reference Context

A reference context is the sheet containing the properties to be used in the evaluation of a function token or reference token.

The reference context can vary for each function token or reference token in a formula expression. The current reference context is the context used for the current token being evaluated.

The CrossPageRef, DocSheetRef, MasterSheetRef, PageSheetRef, ShapeSheetRef, and StyleSheetRef reference tokens specify the reference context of the function token or reference token that immediately follows them in the formula expression. If one of these reference tokens does not precede the token to be evaluated, the default reference context is the sheet containing the formula expression that contains the function token or reference token.

2.2.11.3 Unit Number

A unit number is a numeric value with a unit of measure. Unit numbers represent length, angle, duration and typographic units, higher dimensional forms of these units, and dates.

All unit numbers have a **Dimension** property. One-dimensional unit numbers are used to represent length, angle, duration, and typographic measurements. Two-dimensional units are used to represent area measurements, and three-dimensional units are used to represent volume measurements. A numeric value that has a **Dimension** greater than one is called a multidimensional unit number.

The **Value** property of a one-dimensional unit number is specified as a Custom Internal Unit Type or a date and time as specified by a PtgDate parse token. The **Value** of a multidimensional unit number is specified as a Custom Internal Unit Type for the PtgAcre and PtgHectare operand tokens or as a value as specified by the PtgNumMultiDim parse token.

For numeric values where the **Value** is expressed as a Custom Internal Unit Types, the **Unit** property determines how the numeric value is formatted and displayed in a formula expression, or the user interface. When found in a formula expression or the user interface, the **Value** is converted to the **Unit** and **Dimension** of the operand token; this is called the display value. During formula evaluation, the operand token **Value**, not the display value from the formula expression, is used.

2.2.11.3.1 One-dimensional Unit Number

If the numeric value represents a length or distance measurement, the **Value** property is expressed as a lengthInternalUnitNumber custom internal unit type. The operand tokens that represent length or distance measurements are PtgNumCM, PtgNumDft, PtgNumF, PtgNumFI, PtgNumI, PtgNumKM, PtgNumM, PtgNumMI, PtgNumMM, PtgNumNM, and PtgNumYards.

If the numeric value represents an angle measurement, the **Value** is expressed as an angleInternalUnitNumber custom internal unit type. The operand tokens that represent angles are specified in the vAngle custom token grouping.

If the numeric value represents a duration measurement, the **Value** is expressed as a durationInternalUnitNumber custom internal unit type. The operand tokens that represent durations are PtgEDay, PtgEHour, PtgEMin, PtgESec, PtgEWeek, and PtgTDurDft.

If the numeric value represents a length measurement used in typography, the **Value** is expressed as a typographicInternalUnitNumber custom internal unit type. The operand tokens that represent typographic measurements are PtgTypCD, PtgTypCi, PtgTypDft, PtgTypDi, PtgTypPi, PtgTypPP, and PtgTypPt.

A numeric value with units specified as a PtgPageDft parse token indicates that the internal units are determined by the default values of the drawing page, as specified by PtgPageDft. The **Value** is a number expressed as a Custom Internal Unit Types. For this operand token, the **Unit** of the numeric value is not specified in the PtgPageDft token itself. It is computed as specified by PtgPageDft, and is determined by the default values of the drawing page.

If a numeric value represents a date value, the **Value** is expressed as a date and time of day, in complete extended format, as specified in [ISO-8601] section 4.3.2. The operand token that represents dates is a PtgDate parse token.

The **Value** and **Unit** properties for each unit number are described in the Parse Token Definitions section. The **Dimension** property of one-dimensional unit numbers is equal to one. They do not have a **Currency** property.

2.2.11.3.2 Multidimensional Unit Number

If the numeric value represents an acre or hectare, the **Value** property is expressed as the square of a lengthInternalUnitNumber custom internal unit type. The operand tokens that represent these measurements are PtgAcre and PtgHectare, respectively.

Higher dimensional forms of other unit numbers are specified as a PtgNumMultiDim operand token. The **Value**, **Unit**, and **Dimension** properties of the unit number are stored in the **V** attribute of the Cell_Type element containing the token as specified by the PtgNumMultiDim format.

Multidimensional unit numbers do not have a **Currency** property.

2.3 Parts

The Parts sections that follow specify the structure of the parts that are in the ZIP archive of a web drawing.

2.3.1 Part Enumeration

The web drawing contains the following ZIP package parts and relationships.

Part Name	Relationship between Source and Target Resource	Root Element
App	package	Specified outside this document
Comments	Document	Comments
Connections	Document	DataConnections
Content Type	package	Specified outside this document
Core	package	Specified outside this document
Custom	package	Specified outside this document
Document	package	VisioDocument
Extensions	Document	Extensions
Image	Image or Page	Specified outside this document
Master	Masters	MasterContents
Masters	Document	Masters
Page	Pages	PageContents
Pages	Document	Pages
Recordsets	Document	DataRecordSets
Rels	package	Specified outside this document
Theme	Document	Theme

All other parts are unused and MUST be ignored.

2.3.2 Shared XML Parts and Schema

The Shared XML Parts and Schema sections that follow list the parts in a web drawing that are specified outside this document in their entirety.

2.3.2.1 App XML Part

The App XML part is specified in [ISO/IEC29500-1:2016] section 15.2.12.3.

This is an optional part that specifies the **Extended Properties** of a web drawing, as specified by [ISO/IEC29500-1:2016] section 22.2.

The following properties in the App XML part are defined in a web drawing.

Property Name	Specified in
Application	[ISO/IEC29500-1:2016] section 22.2.2.1
AppVersion	[ISO/IEC29500-1:2016] section 22.2.2.2
Company	[ISO/IEC29500-1:2016] section 22.2.2.5
HeadingPairs	[ISO/IEC29500-1:2016] section 22.2.2.8
HyperlinkBase	[ISO/IEC29500-1:2016] section 22.2.2.11
HyperlinksChanged	[ISO/IEC29500-1:2016] section 22.2.2.12
LinksUpToDate	[ISO/IEC29500-1:2016] section 22.2.2.14
Manager	[ISO/IEC29500-1:2016] section 22.2.2.15
ScaleCrop	[ISO/IEC29500-1:2016] section 22.2.2.22
SharedDoc	[ISO/IEC29500-1:2016] section 22.2.2.23
Template	[ISO/IEC29500-1:2016] section 22.2.2.25
TitlesOfParts	[ISO/IEC29500-1:2016] section 22.2.2.26

2.3.2.2 ContentType XML Part

The ContentType XML part and its syntax are specified in [ISO/IEC29500-2:2012] section 10.1.2.

This part identifies the type of content for each package part.

2.3.2.3 Core XML Part

The Core XML part is specified in [ISO/IEC29500-1:2016] section 15.2.12.1.

This is an optional part that specifies the Core Properties of a web drawing, specified by [ISO/IEC29500-2:2012] section 11.

The following properties in the Core XML part are defined in a web drawing, specified by [ISO/IEC29500-2:2012] Table 11-1.

Property Name
category
created
creator
description
keywords
language
lastModifiedBy
lastPrinted
modified
subject
title

2.3.2.4 Custom XML Part

The Custom XML part is specified in [ISO/IEC29500-1:2016] section 15.2.12.2.

This is an optional part that specifies the Custom Properties of a web drawing, as specified by [ISO/IEC29500-1:2016] section 22.3. The syntax of the Custom Properties is specified by [ISO/IEC29500-1:2016] section 22.3.2.2.

The following properties in the Custom XML Part are defined in a web drawing.

Property Name	Data Type	Data Type Specified in
BuildNumberEdited	i4	[ISO/IEC29500-1:2016] section 22.4.2.14
IsMetric	bool	[ISO/IEC29500-1:2016] section 22.4.2.3

The lower 16 bits of the **BuildNumberEdited** property MUST be greater than 2714.

2.3.2.5 Rels XML Part

The Rels XMP part and its syntax are specified in [ISO/IEC29500-2:2012] section 9.3.

Each set of relationships sharing a common source is represented by XML stored in a Rels XML part.

2.3.3 Visio Parts

The following sections specify the Visio parts that are unique to web drawings and specified in this document.

2.3.3.1 Comments XML Part

An instance of a Comments XML part type that specifies comments in a web drawing. The following properties identify this part:

Content Type:	**application/vnd.ms-visio.comments+xml**
Root Namespace:	http://schemas.openxmlformats.org/officeDocument/2006/relationships
Source Relationship:	http://schemas.microsoft.com/visio/2010/relationships/comments

The Comments XML part MUST be a target of an explicit relationship from a Document XML part. Implicit or explicit relationships to any other parts are unused and MUST be ignored.

The root element for this part MUST be a Comments element.

2.3.3.2 Connections XML Part

An instance of a Connections XML part type that specifies the data connection information needed to query **data sources** and refresh the recordsets referenced by a web drawing. The following properties identify this part:

Content Type:	**application/vnd.ms-visio.connections+xml**
Root Namespace:	http://schemas.microsoft.com/office/visio/2011/1/core
Source Relationship:	http://schemas.microsoft.com/visio/2010/relationships/connections

The Connections XML part MUST be a target of an explicit relationship from a Document XML Part. Implicit or explicit relationships to any other parts are unused and MUST be ignored.

The root element for this part MUST be a DataConnections element.

2.3.3.3 Document XML Part

An instance of a Document XML part type that contains properties of a web drawing. There MUST be exactly one Document XML part in the package. The following properties identify this part:

Content Types:	**application/vnd.ms-visio.drawing.main+xml** **application/vnd.ms-visio.drawing.macroEnabled.main+xml**
Root Namespace:	http://schemas.microsoft.com/office/visio/2011/1/core
Source Relationship:	http://schemas.microsoft.com/visio/2010/relationships/document

The Document XML part MUST be a target of an explicit relationship in the package-relationship item.

The Document XML part is permitted to have explicit relationships to the following parts:

- Connections XML Part
- Masters XML Part
- Pages XML Part
- Recordsets XML Part
- Theme XML Part
- Comments XML Part
- Extensions XML Part

Implicit or explicit relationships to any other parts are unused and MUST be ignored.

The root element for this part MUST be a VisioDocument element.

2.3.3.4 Extensions XML Part

An instance of an Extensions XML part type that specifies extensibility in a web drawing. The following properties identify this part:

Content Type:	application/vnd.ms-visio.extensions+xml
Root Namespace:	http://schemas.microsoft.com/office/visio/2011/1/core
Source Relationship:	http://schemas.microsoft.com/visio/2010/relationships/extensions

The Extensions XML part MUST be a target of an explicit relationship from a Document XML Part. Implicit or explicit relationships to any other parts are unused and MUST be ignored.

The root element for this part MUST be an Extensions element.

2.3.3.5 Image Part

An instance of an Image part type that specifies an image resource used in rendering a web drawing. The following properties identify this part:

Content Type:	**image/bmp** **image/x-emf** **image/gif** **image/jpeg** **image/png** **image/tiff**
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/image

Each part of this type is an image file that conforms to one of the following formats:

- The **bitmap (BMP)** format specified in [MSDN-BMPST].
- The enhanced metafile format (EMF) format specified in [MS-EMF].
- The Graphics Interchange Format (GIF) format specified in [GIF89a].
- The **Joint Photographic Experts Group (JPEG)** format specified in [JFIF].
- The **Portable Network Graphics (PNG)** format specified in [RFC2083].
- The **TIFF** format specified in [RFC3302].

An Image part MUST be a target of an explicit relationship from a Page XML Part except in the case of a fallback image. An Images part MUST NOT have implicit or explicit relationships to any other part specified in this specification.

2.3.3.6 Master XML Part

An instance of a Master XML part type that specifies contents of a master in a web drawing. The following properties identify this part:

Content Type:	application/vnd.ms-visio.master+xml
Root Namespace:	http://schemas.microsoft.com/office/visio/2011/1/core
Source Relationship:	http://schemas.microsoft.com/visio/2010/relationships/master

The Master XML part MUST be a target of an explicit relationship from a Masters part. The Master XML part is permitted to have explicit relationships to the following parts:

- Image Part

Implicit or explicit relationships to any other parts are unused and MUST be ignored except in the case of fallback images.

The root element for this part MUST be a MasterContents element.

2.3.3.7 Masters XML Part

An instance of a Masters XML part type that specifies a collection of masters in a web drawing. The following properties identify this part:

Content Type:	**application/vnd.ms-visio.masters+xml**
Root Namespace:	http://schemas.microsoft.com/office/visio/2011/1/core
Source Relationship:	http://schemas.microsoft.com/visio/2010/relationships/masters

The Masters part MUST be a target of an explicit relationship from a Document XML Part. The Masters part is permitted to have explicit relationships to the following parts:

- Master XML Part

Implicit or explicit relationships to any other parts are unused and MUST be ignored.

The root element for this part MUST be a Masters element.

2.3.3.8 Page XML Part

An instance of a Page XML part type specifies the contents of a drawing page in a web drawing. The following properties identify this part:

Content Type:	**application/vnd.ms-visio.page+xml**
Root Namespace:	http://schemas.microsoft.com/office/visio/2011/1/core
Source Relationship:	http://schemas.microsoft.com/visio/2010/relationships/page

The Page XML part MUST be a target of an explicit relationship from a Pages XML Part. The Page XML part is permitted to have explicit relationships to the following parts:

- Image Part

Implicit or explicit relationships to any other parts are unused and MUST be ignored except in the case of fallback images.

The root element for this part MUST be a PageContents element (section 2.3.4.3.5).

2.3.3.9 Pages XML Part

An instance of a Pages XML part type that specifies a collection of drawing pages in a web drawing. The following properties identify this part:

Content Type:	**application/vnd.ms-visio.pages+xml**
Root Namespace:	http://schemas.microsoft.com/office/visio/2011/1/core
Source Relationship:	http://schemas.microsoft.com/visio/2010/relationships/pages

There MUST be at most one Pages XML part in the package.

The Pages XML part MUST be a target of an explicit relationship from a Document XML Part. The Pages XML part is permitted to have explicit relationships to the following parts:

- Page XML Part

Implicit or explicit relationships to any other parts are unused and MUST be ignored.

The root element for this part MUST be a Pages element.

2.3.3.10 Recordsets XML Part

An instance of a Recordsets XML part type specifies the recordsets and data bindings in a web drawing. The following properties identify this part:

Content Type:	**application/vnd.ms-visio.recordsets+xml**
Root Namespace:	http://schemas.microsoft.com/office/visio/2011/1/core
Source Relationship:	http://schemas.microsoft.com/visio/2010/relationships/recordsets

The Recordsets XML part MUST be a target of an explicit relationship from a Document XML Part. Implicit or explicit relationships to any other parts are unused and MUST be ignored.

The root element for this part MUST be a DataRecordSets element.

2.3.3.11 Theme XML Part

An instance of a Theme XML part type specifies a dynamic theme in a web drawing. The following properties identify this part:

Content Type:	**application/vnd.openxmlformats-officedocument.theme+xml**
Root Namespace:	http://schemas.openxmlformats.org/drawingml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/theme

The Theme XML part MUST be a target of an explicit relationship from a Document XML Part. Implicit or explicit relationships to any other parts are unused and MUST be ignored.

The root element for this part MUST be a Theme element.

2.3.4 Visio XML Schema

The Visio XML Schema sections that follow specifies the XML simple types, complex types, elements and attributes contained in the parts of a web drawing.

2.3.4.1 Simple Types

This specification does not define any simple types.

2.3.4.2 Complex Types

The following Complex Type sections specify the XML complex types contained in the parts of a web drawing.

2.3.4.2.1 AttachedToolbars_Type

Target namespace: http://schemas.microsoft.com/office/visio/2011/1/core

Referenced by: DocumentSettings_Type

A complex type that is unused and MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="AttachedToolbars_Type">
  <xsd:simpleContent>
    <xsd:extension base="xsd:base64Binary"/>
  </xsd:simpleContent>
</xsd:complexType>
```

2.3.4.2.2 AuthorEntry_Type

Target namespace: http://schemas.microsoft.com/office/visio/2011/1/core

Referenced by: AuthorList_Type

A complex type that specifies properties used to identify an author in a Web drawing.

Attributes:

Name: An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the name of the author.

Initials: An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the initials of the author.

ResolutionID: An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that is unused and MUST be ignored.

ID: An xsd:unsignedInt ([XMLSCHEMA2] section 3.3.22) attribute that identifies the author within the Web drawing. It MUST be equal to or greater than one. It MUST be unique amongst all the **ID** attributes of the AuthorEntry_Type child elements of the containing AuthorList_Type element.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="AuthorEntry_Type">
  <xsd:attribute name="Name" type="xsd:string"/>
  <xsd:attribute name="Initials" type="xsd:string"/>
  <xsd:attribute name="ResolutionID" type="xsd:string"/>
  <xsd:attribute name="ID" type="xsd:unsignedInt" use="required"/>
</xsd:complexType>
```

2.3.4.2.3 AuthorList_Type

Target namespace: http://schemas.microsoft.com/office/visio/2011/1/core

Referenced by: Comments_Type

A complex type that specifies the authors in a web drawing.

Child Elements:

AuthorEntry: An AuthorEntry_Type element that specifies properties used to identify an author in a web drawing.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="AuthorList_Type">
  <xsd:sequence>
    <xsd:element name="AuthorEntry" type="AuthorEntry_Type" minOccurs="0"
maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```


2.3.4.2.4 AutoLinkComparison_Type

Target namespace: http://schemas.microsoft.com/office/visio/2011/1/core

Referenced by: DataRecordSet_Type

A complex type that is unused and MUST be ignored.

Attributes:

ColumnName: An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that is unused and MUST be ignored.

ContextType: An xsd:unsignedInt ([XMLSCHEMA2] section 3.3.22) attribute that is unused and MUST be ignored.

ContextTypeLabel: An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that is unused and MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="AutoLinkComparison_Type">
  <xsd:attribute name="ColumnName" type="xsd:string" use="required"/>
  <xsd:attribute name="ContextType" type="xsd:unsignedInt" use="required"/>
  <xsd:attribute name="ContextTypeLabel" type="xsd:string"/>
</xsd:complexType>
```

2.3.4.2.5 Cell_Type

Target namespace: http://schemas.microsoft.com/office/visio/2011/1/core

Referenced by: StyleSheet_Type, Section_Type, Row_Type, Sheet_Type, DocumentSheet_Type, PageSheet_Type, ShapeSheet_Type

A complex type that specifies a single property, which can also be used to represent an operand token.

Child Elements:

RefBy: A complex type that is unused and MUST be ignored.

Attributes:

N: An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the language-independent name of the property. It MUST be unique amongst all of the Cell_Type elements of the containing Row_Type element, and MUST be equal to a value specified in the Cells (section 2.4.4) section of this specification.

U: An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies how this property is formatted and displayed in a user interface, and how it is used in a formula expression. If present, it MUST be equal to a value from the following table.

Value	Meaning
AC	Acres
DEG	Degrees
DA	Radians
AD	Degrees-minutes-seconds
RAD	Radians
BOOL	Boolean
COLOR	**RGB** color value
CY	Currency
DATE	Days
ED	Days
EH	Hours
EM	Minutes
ES	Seconds
EW	Weeks
HA	Hectare
CM	Centimeters
DL	Inches
FT	Feet
F_I	Feet and inches
IN	Inches
IN_F	Inches
KM	Kilometers
M	Meters
MI	Miles
MI_F	Miles
MM	Millimeters
NM	Nautical miles
PER	Percentage
YD	Yards
DP	Inches
PNT	Coordinates of a two-dimensional point
STR	String

Value	Meaning
DE	Ddots
C_D	Ciceros and didots
C	Ciceros
D	Didots
DT	Points
P	Picas
P_PT	Picas and points
PT	Points

E: An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the error state of the property, obtained during a formula evaluation. If present, it MUST be equal to a value from the following table.

Value	Meaning
#DIM!	An error value that specifies that a dimensional value exceeds the dimension range.
#DIV/0!	An error value that specifies division by zero.
#VALUE!	An error value that specifies that an operand token is of the wrong type.
#REF!	An error value that specifies that a reference to a cell does not exist.
#NUM!	An error value that specifies an invalid number.
#N/A	An error value that specifies that a value is not available.

F: An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the formula expression of the property. It MUST be either a formula expression that satisfies the Formula ABNF and Full Grammar Definition in this specification or equal to a value in the following table.

Value	Meaning
No Formula	Specifies that no formula exists.
Inh	Specifies a formula that is inherited.

V: An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the value of the property. It MUST be equal to "1.#INF" if it specifies a **floating-point number** that is larger than 1.7976e308. If the value of the **V** attribute is equal to "themed", the value of the property is specified by theme inheritance.

When the **F** attribute is present, the value of the **V** attribute MUST be used until a formula evaluation is triggered on the **F** attribute that does not result in an error value. After formula evaluation is triggered on the **F** attribute, the value of the property is specified by the most recent result of the formula evaluation that does not produce an error value.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.


```
<xsd:complexType name="Cell_Type" mixed="true">
  <xsd:sequence>
    <xsd:element name="RefBy" type="RefBy Type" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="N" type="xsd:string" use="required"/>
  <xsd:attribute name="U" type="xsd:string"/>
  <xsd:attribute name="E" type="xsd:string"/>
  <xsd:attribute name="F" type="xsd:string"/>
  <xsd:attribute name="V" type="xsd:string"/>
</xsd:complexType>
```

2.3.4.2.6 CellDef_Type

Target namespace: http://schemas.microsoft.com/office/visio/2011/1/core

Referenced by: Extensions_Type, SectionDef_Type, RowDef_Type

A complex type that specifies the definition of a cell that is not specified in this specification.

Attributes:

N: An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the name of the cell. It MUST be unique amongst all the FunctionDef_Type, CellDef_Type, and SectionDef_Type elements in the Web drawing. It MUST NOT be equal to the name of a function token listed in the Function Token Definitions section of this specification. It MUST NOT be equal to the name of a section listed in the Sections section of this specification. It MUST NOT be equal to the name of a cell listed in the Cells section of this specification.

T: An xsd:token ([XMLSCHEMA2] section 3.3.2) attribute that specifies the operand token used to specify the **Value** of the cell. It MUST be equal to a value from the following table.

Value	Operand Token
BYTE	PtgByte
BOOL	PtgBool
WORD	PtgUnsShort
SHORT	PtgShort
LONG	PtgInt
DOUBLE	PtgNum
PERCENT	PtgNum
MULTIDIM	PtgNumMultiDim
CAL	vCalendar

F: An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the default formula expression of the cell.

IX: An xsd:unsignedByte ([XMLSCHEMA2] section 3.3.24) attribute that specifies the zero-based identifier of a collection of cells. It MUST be unique amongst all of the CellDef_Type elements of the

containing RowDef_Type element. It MUST be greater than the **IX** attribute of any preceding CellDef_Type element of the containing Extensions_Type, SectionDef_Type or RowDef_Type element. If the containing element is a RowDef_Type element and if the RowDef_Type element's containing element is a SectionDef_Type element with **T** attribute equal to "Indexed" or **N** attribute equal to "Character", "Field", "FillGradient", "Geometry", "Layer", "LineGradient", "Paragraph", "Reviewer", "Scratch", or "Tabs", **IX** MUST exist.

S: An xsd:unsignedByte ([XMLSCHEMA2] section 3.3.24) attribute that is unused and MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CellDef_Type">
  <xsd:attribute name="N" type="xsd:string" use="required"/>
  <xsd:attribute name="T" type="xsd:token" use="required"/>
  <xsd:attribute name="F" type="xsd:string"/>
  <xsd:attribute name="IX" type="xsd:unsignedByte"/>
  <xsd:attribute name="S" type="xsd:unsignedByte"/>
</xsd:complexType>
```

2.3.4.2.7 ColorEntry_Type

Target namespace: http://schemas.microsoft.com/office/visio/2011/1/core

Referenced by: Colors_Type

A complex type that specifies a color available in a color table.

Attributes:

IX: An xsd:unsignedInt ([XMLSCHEMA2] section 3.3.22) attribute that specifies the **zero-based index** of the element. It MUST be less than or equal to 253. It MUST be unique amongst all of the ColorEntry_Type elements of the containing Colors_Type.

RGB: An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the hexadecimal value of a color in the color table.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="ColorEntry_Type">
  <xsd:attribute name="IX" type="xsd:unsignedInt" use="required"/>
  <xsd:attribute name="RGB" type="xsd:string" use="required"/>
</xsd:complexType>
```

2.3.4.2.8 Colors_Type

Target namespace: http://schemas.microsoft.com/office/visio/2011/1/core

Referenced by: VisioDocument_Type

A complex type that specifies the color table of a web drawing.

Child Elements:

ColorEntry: A ColorEntry_Type element that specifies the colors available in a color table.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.


```
<xsd:complexType name="Colors_Type">
  <xsd:sequence>
    <xsd:element name="ColorEntry" type="ColorEntry Type" minOccurs="1"
maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

2.3.4.2.9 CommentEntry_Type

Target namespace: http://schemas.microsoft.com/office/visio/2011/1/core

Referenced by: CommentList_Type

A complex type that specifies properties used to identify a comment in a web drawing.

Attributes:

AuthorID: An xsd:unsignedInt ([XMLSCHEMA2] section 3.3.22) attribute that is a value that identifies the author. It MUST be equal to or greater than one.

PageID: An xsd:unsignedInt ([XMLSCHEMA2] section 3.3.22) attribute that is a value that identifies the drawing page the comment is on. The comment MUST be contained in the drawing page specified by **PageID.**

ShapeID: An xsd:unsignedInt ([XMLSCHEMA2] section 3.3.22) attribute that is a value that identifies the shape the comment is on. If no **ShapeID** is specified, the comment refers to the drawing page.

Date: An xsd:dateTime ([XMLSCHEMA2] section 3.2.7) attribute that specifies when a comment was created.

EditDate: An xsd:dateTime ([XMLSCHEMA2] section 3.2.7) attribute that specifies when a comment was last changed. The **EditDate** MUST be greater than or equal to the value of **Date**.

Done: An xsd:boolean ([XMLSCHEMA2] section 3.2.2) attribute that specifies the current state of the comment. It MUST be equal to zero or one.

CommentID: An xsd:unsignedInt ([XMLSCHEMA2] section 3.3.22) attribute that is a unique value that identifies the comment in a drawing page. It MUST be unique amongst all the **CommentEntry_Type** child elements of the containing CommentList_Type.

AutoCommentType: An xsd:unsignedInt ([XMLSCHEMA2] section 3.3.22) attribute that is unused and MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CommentEntry Type">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="AuthorID" type="xsd:unsignedInt" use="required"/>
      <xsd:attribute name="PageID" type="xsd:unsignedInt" use="required"/>
      <xsd:attribute name="ShapeID" type="xsd:unsignedInt"/>
      <xsd:attribute name="Date" type="xsd:dateTime" use="required"/>
      <xsd:attribute name="EditDate" type="xsd:dateTime"/>
      <xsd:attribute name="Done" type="xsd:boolean"/>
      <xsd:attribute name="CommentID" type="xsd:unsignedInt" use="required"/>
      <xsd:attribute name="AutoCommentType" type="xsd:unsignedInt"/>
    </xsd:extension>
```

```
  </xsd:simpleContent>
</xsd:complexType>
```


2.3.4.2.10 CommentList_Type

Target namespace: http://schemas.microsoft.com/office/visio/2011/1/core

Referenced by: Comments_Type

A complex type that specifies the comments in a web drawing.

Child Elements:

CommentEntry: A CommentEntry_Type element that specifies properties used to identify a comment in a web drawing.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CommentList_Type">
  <xsd:sequence>
    <xsd:element name="CommentEntry" type="CommentEntry_Type" minOccurs="0"
maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

2.3.4.2.11 Comments_Type

Target namespace: http://schemas.microsoft.com/office/visio/2011/1/core

Referenced by: Comments

A complex type that specifies properties used to identify the authors and comments in a web drawing.

Child Elements:

AuthorList: An AuthorList_Type element that specifies the authors in a web drawing.

CommentList: A CommentList_Type element that specifies the comments in a web drawing.

Attributes:

ShowCommentTags: An xsd:boolean ([XMLSCHEMA2] section 3.2.2) attribute that is unused and MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="Comments_Type">
  <xsd:sequence>
    <xsd:element name="AuthorList" type="AuthorList Type" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="CommentList" type="CommentList_Type" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="ShowCommentTags" type="xsd:boolean"/>
</xsd:complexType>
```

2.3.4.2.12 Connect_Type

Target namespace: http://schemas.microsoft.com/office/visio/2011/1/core

Referenced by: Connects_Type

A complex type that is unused and MUST be ignored.

Attributes:

FromSheet: An xsd:unsignedInt ([XMLSCHEMA2] section 3.3.22) attribute that is unused and MUST be ignored.

FromCell: An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that is unused and MUST be ignored.

FromPart: An xsd:int ([XMLSCHEMA2] section 3.3.17) attribute that is unused and MUST be ignored.

ToSheet: An xsd:unsignedInt ([XMLSCHEMA2] section 3.3.22) attribute that is unused and MUST be ignored.

ToCell: An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that is unused and MUST be ignored.

ToPart: An xsd:int ([XMLSCHEMA2] section 3.3.17) attribute that is unused and MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="Connect_Type">
  <xsd:attribute name="FromSheet" type="xsd:unsignedInt" use="required"/>
  <xsd:attribute name="FromCell" type="xsd:string"/>
  <xsd:attribute name="FromPart" type="xsd:int"/>
  <xsd:attribute name="ToSheet" type="xsd:unsignedInt" use="required"/>
  <xsd:attribute name="ToCell" type="xsd:string"/>
  <xsd:attribute name="ToPart" type="xsd:int"/>
</xsd:complexType>
```

2.3.4.2.13 Connects_Type

Target namespace: http://schemas.microsoft.com/office/visio/2011/1/core

Referenced by: PageContents_Type

A complex type that is unused and MUST be ignored.

Child Elements:

Connect: A Connect_Type element that is unused and MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="Connects_Type">
  <xsd:sequence>
    <xsd:element name="Connect" type="Connect Type" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

2.3.4.2.14 cp_Type

Target namespace: http://schemas.microsoft.com/office/visio/2011/1/core

Referenced by: Text_Type

A complex type that specifies the beginning of a text run, and specifies an index designating the set of character properties to use.

Attributes:

IX: An xsd:unsignedInt ([XMLSCHEMA2] section 3.3.22) attribute that specifies the character properties used in the **text run**. It MUST be the **IX** attribute of a Row_Type that has a Character Section_Type parent element.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="cp_Type">
  <xsd:attribute name="IX" type="xsd:unsignedInt" use="required"/>
</xsd:complexType>
```

2.3.4.2.15 CT_FmtSchemeEx

Target namespace: http://visThemeSchemaUri

Referenced by: **Ext** element as specified by the **CT_OfficeArtExtension** type specified in [ISO/IEC29500-1:2016] section 20.1.2.2.14.

Child Elements:

schemeID: A CT_SchemeID element that specifies the index of an effect scheme dynamic theme component or a connector scheme dynamic theme component.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_FmtSchemeEx" oxsd:cname="StyleMatrixEx" oxsd:cwrap="noTemplate">
  <xsd:sequence>
    <xsd:element name="schemeID" type="CT_SchemeID" minOccurs="1" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>
```

2.3.4.2.16 CT_FontProps

Target namespace: http://visThemeSchemaUri

Referenced by: CT_FontStyles

Specifies properties used to format a **text run**.

Attributes:

style: An xsd:unsignedInt ([XMLSCHEMA2] section 3.3.22) attribute that specifies properties used to format a text run. The value of the structure MUST be a bitwise OR combination of one or more of the values from the table in the Style Cell_Type element.

Child Elements:

color: A **CT_Color** type specified in [ISO/IEC29500-1:2016] section A.2 that specifies color properties used to format a text run.

extLst: An a:CT_OfficeArtExtensionList ([ISO/IEC29500-1:2016] section 20.1.2.2.15) type which is unused and MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.


```
<xsd:complexType name="CT_FontProps" oxsd:cname="FontProps" oxsd:cwrap="noTemplate">
  <xsd:sequence>
    <xsd:element name="color" type="a:CT Color" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="extLst" oxsd:cname="ext" type="a:CT_OfficeArtExtensionList"
minOccurs="0" maxOccurs="1" oxsd:dataStructure="optional"/>
  </xsd:sequence>
  <xsd:attribute name="style" type="xsd:unsignedInt" use="required" oxsd:cname="style"/>
</xsd:complexType>
```

2.3.4.2.17 CT_FontStyles

Target namespace: http://visThemeSchemaUri

Referenced by: CT_FontStylesGroup

Specifies a set of properties used to format a **text run**.

Child Elements:

fontProps: A CT_FontProps element that specifies properties used to format a text run.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_FontStyles" oxsd:cname="FontStyles" oxsd:cwrap="noTemplate">
    <xsd:sequence>
      <xsd:element name="fontProps" oxsd:cname="fontProps" type="CT_FontProps" minOccurs="3"
maxOccurs="unbounded" />
    </xsd:sequence>
</xsd:complexType>
```

2.3.4.2.18 CT_FontStylesGroup

Target namespace: http://visThemeSchemaUri

Referenced by: **Ext** element as specified by the **CT_OfficeArtExtension** type specified in [ISO/IEC29500-1:2016] section 20.1.2.2.14.

Specifies the properties used to format a **text run** in shapes.

Child Elements:

connectorFontStyles: A CT_FontStyles element that specifies the properties used to format a text run in a connector shape.

fontStyles: A CT_FontStyles element that specifies the properties used to format a text run in a non-connector shape.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_FontStylesGroup" oxsd:cname="FontStylesGroup"
oxsd:cwrap="noTemplate">

    <xsd:sequence>
      <xsd:element name="connectorFontStyles" oxsd:cname="connectorFontStyles"
type="CT_FontStyles" minOccurs="1" maxOccurs="1" />
```

```
      <xsd:element name="fontStyles" oxsd:cname="fontStyles" type="CT FontStyles"
minOccurs="1" maxOccurs="1" />
    </xsd:sequence>

</xsd:complexType>
```


2.3.4.2.19 CT_LineEx

Target namespace: http://visThemeSchemaUri

Referenced by: CT_LineStyle

A complex type that specifies line properties information of an effect scheme or a connector scheme dynamic theme component in a dynamic theme.

Attributes:

rndg: An a:ST_PositiveCoordinate ([ISO/IEC29500-1:2016] section 20.1.10.42) attribute that specifies the rounding radius of the outline of a shape. The value of the structure MUST be greater than or equal to zero inches. The value of zero specifies that there is no rounding. A value greater than zero specifies that any corner between two line segments, a line segment and an elliptical arc, or two elliptical arcs within the outline is rounded with a radius equal to the value.

start: An xsd:unsignedByte ([XMLSCHEMA2] section 3.3.24) attribute that specifies an arrowhead at the first vertex of a one-dimensional shape.

The value of the structure MUST be specified by the table in the BeginArrow Cell_Type element, and it MUST NOT be 254.

startSize: An xsd:unsignedByte ([XMLSCHEMA2] section 3.3.24) attribute that specifies the size of the arrowhead at the first vertex of a shape.

The value of the structure MUST be specified by the table in the BeginArrowSize Cell_Type element.

end: An xsd:unsignedByte ([XMLSCHEMA2] section 3.3.24) attribute that specifies an arrowhead at the last vertex of a one-dimensional shape.

The value of the structure MUST be specified by the table in the BeginArrow Cell_Type element, and it MUST NOT be 254.

endSize: An xsd:unsignedByte ([XMLSCHEMA2] section 3.3.24) attribute that specifies the size of the arrowhead at the last vertex of a shape.

The value of the structure MUST be specified by the table in the BeginArrowSize Cell_Type element.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_LineEx" oxsd:cname="LineEx" oxsd:cwrap="noTemplate"
oxsd:cexport="true">
  <xsd:attribute name="rndg" type="a:ST PositiveCoordinate" use="optional"
oxsd:cname="rounding"/>
  <xsd:attribute name="start" type="xsd:unsignedByte" use="optional"
oxsd:cname="startSymbol"/>
  <xsd:attribute name="startSize" type="xsd:unsignedByte" use="optional"
oxsd:cname="startSymbolSize"/>
  <xsd:attribute name="end" type="xsd:unsignedByte" use="optional" oxsd:cname="endSymbol"/>
  <xsd:attribute name="endSize" type="xsd:unsignedByte" use="optional"
oxsd:cname="endSymbolSize"/>
</xsd:complexType>
```

2.3.4.2.20 CT_LineStyle

Target namespace: http://visThemeSchemaUri

Referenced by: CT_SchemeLineStyles

Specifies line properties and sketch effect set information of an effect scheme dynamic theme component or a connector scheme dynamic theme component in a dynamic theme.

Child Elements:

lineEx: A CT_LineEx element that specifies the line properties information.

sketch: A CT_Sketch element that specifies sketch effect set information.

extLst: An a:CT_OfficeArtExtensionList ([ISO/IEC29500-1:2016] section 20.1.2.2.15) type which is unused and MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_LineStyle" oxsd:cname="LineStyle" oxsd:cwrap="noTemplate">
  <xsd:sequence>
    <xsd:element name="lineEx" oxsd:cname="lineEx" type="CT_LineEx" minOccurs="1"
maxOccurs="1"/>
    <xsd:element name="sketch" oxsd:cname="sketch" type="CT_Sketch" minOccurs="0"
maxOccurs="1" oxsd:dataStructure="optional"/>
    <xsd:element name="extLst" oxsd:cname="ext" type="a:CT_OfficeArtExtensionList"
minOccurs="0" maxOccurs="1" oxsd:dataStructure="optional"/>
  </xsd:sequence>
</xsd:complexType>
```

2.3.4.2.21 CT_LineStyles

Target namespace: http://visThemeSchemaUri

Referenced by: **Ext** element as specified by the **CT_OfficeArtExtension** type specified in [ISO/IEC29500-1:2016] section 20.1.2.2.14.

Child Elements:

fmtConnectorSchemeLineStyles: A CT_SchemeLineStyles element that specifies line properties and sketch effect set information of a connector scheme dynamic theme component in a dynamic theme.

fmtSchemeLineStyles: A CT_SchemeLineStyles element that specifies line properties and sketch effect set information of an effect scheme dynamic theme component in a dynamic theme.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_LineStyles" oxsd:cname="LineStyles" oxsd:cwrap="noTemplate">
  <xsd:sequence>
    <xsd:element name="fmtConnectorSchemeLineStyles"
oxsd:cname="fmtConnectorSchemeLineStyles" type="CT SchemeLineStyles" minOccurs="1"
maxOccurs="1"/>
    <xsd:element name="fmtSchemeLineStyles" oxsd:cname="fmtSchemeLineStyles"
type="CT_SchemeLineStyles" minOccurs="1" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>
```

2.3.4.2.22 CT_OfficeStyleSheet

Target namespace: http://schemas.openxmlformats.org/drawingml/2006/main

Referenced by: Theme

A complex type specified in [ISO/IEC29500-1:2016] section 20.1.6.9 that specifies a dynamic theme.

The following descendant elements of a **CT_OfficeStyleSheet** element specified in [ISO/IEC29500-1:2016] section 20.1.6.9 are unused and MUST be ignored.

- **ObjectDefaults** element detailed by the **CT_ObjectStyleDefaults** type specified in [ISO/IEC29500-1:2016] section 20.1.6.7.
- **ExtraClrSchemeLst** element detailed by the **CT_ColorSchemeList** type specified in [ISO/IEC29500-1:2016] section 20.1.6.5.
- **Dk2** element detailed by the **CT_Color** type specified in [ISO/IEC29500-1:2016] section 20.1.4.1.10.
- **Lt2** element detailed by the **CT_Color** type specified in [ISO/IEC29500-1:2016] section 20.1.4.1.23.
- **Hlink** element detailed by the **CT_Color** type specified in [ISO/IEC29500-1:2016] section 20.1.4.1.19.
- **FolHlink** element detailed by the **CT_Color** type specified in [ISO/IEC29500-1:2016] section 20.1.4.1.15.
- **MajorFont** element detailed by the **CT_FontCollection** type specified in [ISO/IEC29500-1:2016] section 20.1.4.1.24.
- **BgFillStyleLst** element detailed by the **CT_BackgroundFillStyleList** type specified in [ISO/IEC29500-1:2016] section 20.1.4.1.7.
- **Camera** element detailed by the **CT_Camera** type specified in [ISO/IEC29500-1:2016] section 20.1.5.5.
- **HueMod** element detailed by the **CT_PositivePercentage** type specified in [ISO/IEC29500-1:2016] section 20.1.2.3.15.
- **CustClrLst** element detailed by the **CT_CustomColorList** type specified in [ISO/IEC29500-1:2016] section 20.1.6.3.
- **HeadEnd** element detailed by the **CT_LineEndProperties** type specified in [ISO/IEC29500-1:2016] section 20.1.8.38.
- **TailEnd** element detailed by the **CT_LineEndProperties** type specified in [ISO/IEC29500-1:2016] section 20.1.8.57.
- **Round** element detailed by the **CT_LineJoinRound** type specified in [ISO/IEC29500-1:2016] section 20.1.8.52.
- **PattFill** element detailed by the **CT_PatternFillProperties** type specified in [ISO/IEC29500-1:2016] section 20.1.8.47.
- **NoFill** element detailed by the **CT_NoFillProperties** type specified in [ISO/IEC29500-1:2016] section 20.1.8.44.
- **Miter** element detailed by the **CT_LineJoinMiterProperties** type specified in [ISO/IEC29500-1:2016] section 20.1.8.43.

- **CustDash** element detailed by the **CT_DashStopList** type specified in [ISO/IEC29500-1:2016] section 20.1.8.21.

- **BlipFill** element detailed by the **CT_BlipFillProperties** type specified in [ISO/IEC29500-1:2016] section 20.1.8.14.
- **GrpFill** element detailed by the **CT_GroupFillProperties** type specified in [ISO/IEC29500-1:2016] section 20.1.8.35.
- **TileRect** element detailed by the **CT_RelativeRect** type specified in [ISO/IEC29500-1:2016] section 20.1.8.59.
- **EffectDag** element detailed by the **CT_EffectContainer** type specified in [ISO/IEC29500-1:2016] section 20.1.8.25.
- **Blur** element detailed by the **CT_BlurEffect** type specified in [ISO/IEC29500-1:2016] section 20.1.8.15.
- **FillOverlay** element detailed by the **CT_FillOverlayEffect** type specified in [ISO/IEC29500-1:2016] section 20.1.8.29.
- **PrstShdw** element detailed by the **CT_PresetShadowEffect** type specified in [ISO/IEC29500-1:2016] section 20.1.8.49.

The attributes of the descendant elements of a **CT_OfficeStyleSheet** element specified in [ISO/IEC29500-1:2016] section 20.1.6.9 listed in the following table are unused and MUST be ignored.

Element	Attributes
Lin as specified by the **CT_LinearShadeProperties** type specified in [ISO/IEC29500-1:2016] section 20.1.8.41.	**Scaled**
Ln as specified by the **CT_LineProperties** type specified in [ISO/IEC29500-1:2016] section 20.1.2.2.24.	**Algn**
LightRig as specified by the **CT_LightRig** type specified in [ISO/IEC29500-1:2016] section 20.1.5.9.	**Dir**
Rot as specified by the **CT_SphereCoords** type specified in [ISO/IEC29500-1:2016] section 20.1.5.11.	**Lat** and **long**
GradFill as specified by the **CT_GradientFillProperties** type specified in [ISO/IEC29500-1:2016] section 20.1.8.33.	**Flip**
OuterShdw as specified by the **CT_OuterShadowEffect** type specified in [ISO/IEC29500-1:2016] section 20.1.8.45.	**Algn**, **kx**, **ky**, **sx**, and **sy**
Reflection as specified by the **CT_ReflectionEffect** type specified in [ISO/IEC29500-1:2016] section 20.1.8.50.	**Algn**, **dir**, **endA**, **fadeDir**, **kx**, **ky**, **rotWithShape**, **stPos**, **sx**, and **sy**

Child Elements:

themeElements element detailed by the **CT_BaseStyles** type specified in [ISO/IEC29500-1:2016] section §A.4.1. This element specifies the dynamic theme components of a dynamic theme.

objectDefaults element detailed by the **CT_ObjectStyleDefaults** type specified in [ISO/IEC29500-1:2016] section 20.1.6.7.

extraClrSchemeLst element detailed by the **CT_ColorSchemeList** type specified in[ISO/IEC29500-1:2016] section 20.1.6.5.

custClrLst element detailed by the **CT_CustomColorList** type specified in [ISO/IEC29500-1:2016] section 20.1.6.3.

extLst element detailed by the **CT_OfficeArtExtensionList** type specified in [ISO/IEC29500-1:2016] section 20.1.2.2.15.

Attributes:

name: An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the language-independent name of the dynamic theme.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_OfficeStyleSheet">
  <xsd:sequence>
    <xsd:element name="themeElements" type="CT_BaseStyles" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="objectDefaults" type="CT_ObjectStyleDefaults" minOccurs="0"
maxOccurs="1"/>
    <xsd:element name="extraClrSchemeLst" type="CT_ColorSchemeList" minOccurs="0"
maxOccurs="1"/>
    <xsd:element name="custClrLst" type="CT_CustomColorList" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string" use="optional" default=""/>
</xsd:complexType>
```

2.3.4.2.23 CT_SchemeID

Target namespace: http://visThemeSchemaUri

Referenced by: CT_ThemeScheme, CT_FmtSchemeEx, and **ext** element detailed by the **CT_OfficeArtExtension** type specified in [ISO/IEC29500-1:2016] section 20.1.2.2.14.

Specifies the index of a color scheme, font scheme, effect scheme, connector scheme, or primary scheme dynamic theme component in a dynamic theme, or the GUID of a custom dynamic theme color scheme.

Attributes:

schemeEnum: An xsd:unsignedInt ([XMLSCHEMA2] section 3.3.22) attribute that specifies the one-based index of a color scheme, font scheme, effect scheme, connector scheme, or primary scheme dynamic theme component. If the value of the structure is equal to 65535, the GUID of a custom dynamic theme color scheme is specified by the **schemeGUID** attribute.

schemeGUID: An a:ST_Guid ([ISO/IEC29500-1:2016] section 22.9.2.4) attribute that specifies the GUID of a custom dynamic theme color scheme. If the value of the **schemeEnum** attribute is not equal to 65535, this attribute is unused and MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_SchemeID" oxsd:cname="SchemeID" oxsd:cwrap="noTemplate">
  <xsd:attribute name="schemeEnum" type="xsd:unsignedInt" use="optional" oxsd:cname="enum"/>
  <xsd:attribute name="schemeGUID" type="a:ST_Guid" use="optional" oxsd:cname="guid"/>
```

```
</xsd:complexType>
```


2.3.4.2.24 CT_SchemeLineStyles

Target namespace: http://visThemeSchemaUri

Referenced by: CT_LineStyles

Specifies a set of line properties and sketch effect set information of an effect scheme or connector scheme dynamic theme component in a dynamic theme.

Child Elements:

lineStyle: A CT_LineStyle element that specifies line properties and sketch effect set information of an effect scheme or connector scheme dynamic theme component.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_SchemeLineStyles" oxsd:cname="SchemeLineStyles"
oxsd:cwrap="noTemplate">
  <xsd:sequence>
    <xsd:element name="lineStyle" oxsd:cname="lineStyle" type="CT_LineStyle" minOccurs="3"
maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

2.3.4.2.25 CT_Sketch

Target namespace: http://visThemeSchemaUri

Referenced by: CT_LineStyle

A complex type that specifies sketch effect set information of an effect scheme or connector scheme dynamic theme component in a dynamic theme.

Attributes:

lnAmp: An a:ST_PositiveFixedPercentage ([ISO/IEC29500-1:2016] section 22.9.2.10) attribute that specifies the amplitude of the path perturbations for a sketch effect set. The value of the structure MUST be expressed as a percentage, and MUST be greater than or equal to zero and less than or equal to one. The value is normalized such that a value of 1 corresponds to 100 percent. A value of zero specifies no perturbation to the path; a value of one specifies maximum perturbation.

fillAmp: An a:ST_PositiveFixedPercentage ([ISO/IEC29500-1:2016] section 22.9.2.10) attribute that specifies the amplitude of the fill perturbations for a sketch effect set. The value of the structure MUST be expressed as a percentage, and MUST be greater than or equal to zero and less than or equal to one. The value is normalized such that a value of 1 corresponds to 100 percent. A value of zero specifies no perturbation to the fill; a value of one specifies maximum perturbation.

lnWeight: An a:ST_PositiveCoordinate ([ISO/IEC29500-1:2016] section 22.1.10.42) attribute that specifies the amplitude of the path perturbations for a sketch effect set. The value of the structure MUST be expressed as a percentage, and MUST be greater than or equal to zero and less than or equal to one. The value is normalized such that a value of 1 corresponds to 100 percent. A value of zero specifies no perturbation to the path; a value of one specifies maximum perturbation.

numPts: An xsd:unsignedByte ([XMLSCHEMA2] section 3.3.24) attribute that specifies the number of points, distributed uniformly across each path segment of a shape, where perturbations are performed

for a sketch effect set. It MUST have a value greater than or equal to zero and less than or equal to 25, with a default value of five.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_Sketch" oxsd:cname="Sketch" oxsd:cwrap="noTemplate"
oxsd:cexport="true">
  <xsd:attribute name="lnAmp" type="a:ST_PositiveFixedPercentage" use="optional"
oxsd:cname="lnAmp"/>
  <xsd:attribute name="fillAmp" type="a:ST_PositiveFixedPercentage" use="optional"
oxsd:cname="fillAmp"/>
  <xsd:attribute name="lnWeight" type="a:ST_PositiveCoordinate" use="optional"
oxsd:cname="lnWeight"/>
  <xsd:attribute name="numPts" type="xsd:unsignedByte" use="optional" oxsd:cname="numPts"/>
</xsd:complexType>
```

2.3.4.2.26 CT_ThemeScheme

Target namespace: http://visThemeSchemaUri

Referenced by: **Ext** element as detailed by the **CT_OfficeArtExtension** type specified in [ISO/IEC29500-1:2016] section 20.1.2.2.14.

Specifies the primary scheme dynamic theme component in a dynamic theme.

Child Elements:

schemeID: A CT_SchemeID element that specifies the index of the primary scheme dynamic theme component.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_ThemeScheme" oxsd:cname="ThemeScheme" oxsd:cwrap="noTemplate">
  <xsd:sequence>
    <xsd:element name="schemeID" type="CT_SchemeID" minOccurs="1" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>
```

2.3.4.2.27 CT_VarClrScheme

Target namespace: http://visThemeSchemaUri

Referenced by: CT_VariationClrSchemeLst

Specifies a color scheme list of a dynamic theme variant.

Attributes:

monotone: An xsd:boolean ([XMLSCHEMA2] section 3.2.2) attribute that specifies multiformat information of a dynamic theme component in a dynamic theme. True if scheme is monotone; False otherwise.

Child Elements:

VarColor1: A **CT_Color** type specified in [ISO/IEC29500-1:2016] section A.2 that specifies a color property.

VarColor2: A **CT_Color** type specified in [ISO/IEC29500-1:2016] section A.2 that specifies a color property.

VarColor3: A **CT_Color** type specified in [ISO/IEC29500-1:2016] section A.2 that specifies a color property.

VarColor4: A **CT_Color** type specified in [ISO/IEC29500-1:2016] section A.2 that specifies a color property.

VarColor5: A **CT_Color** type specified in [ISO/IEC29500-1:2016] section A.2 that specifies a color property.

VarColor6: A **CT_Color** type specified in [ISO/IEC29500-1:2016] section A.2 that specifies a color property.

VarColor7: A **CT_Color** type specified in [ISO/IEC29500-1:2016] section A.2 that specifies a color property.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_VarClrScheme" oxsd:cname="VariationColorScheme"
oxsd:cwrap="noTemplate">
    <xsd:sequence>
      <xsd:element name="varColor1" type="a:CT_Color" minOccurs="1" maxOccurs="1"/>
      <xsd:element name="varColor2" type="a:CT_Color" minOccurs="1" maxOccurs="1"/>
      <xsd:element name="varColor3" type="a:CT_Color" minOccurs="1" maxOccurs="1"/>
      <xsd:element name="varColor4" type="a:CT_Color" minOccurs="1" maxOccurs="1"/>
      <xsd:element name="varColor5" type="a:CT_Color" minOccurs="1" maxOccurs="1"/>
      <xsd:element name="varColor6" type="a:CT_Color" minOccurs="1" maxOccurs="1"/>
      <xsd:element name="varColor7" type="a:CT_Color" minOccurs="1" maxOccurs="1"/>
      <xsd:element name="extLst" oxsd:cname="ext" type="a:CT_OfficeArtExtensionList"
minOccurs="0" maxOccurs="1" oxsd:dataStructure="optional"/>
    </xsd:sequence>
    <xsd:attribute name="monotone" type="xsd:boolean" use="optional" default="false"/>
  </xsd:complexType>
```

2.3.4.2.28 CT_VariationClrSchemeLst

Target namespace: http://visThemeSchemaUri

Referenced by: **Ext** element as detailed by the **CT_OfficeArtExtension** type specified in [ISO/IEC29500-1:2016] section 20.1.2.2.14.

Specifies four distinct color scheme lists of four distinct dynamic theme variants in a dynamic theme.

Child Elements:

VariationClrScheme: A CT_VarClrScheme type that specifies a color scheme list of a dynamic theme variant.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_VariationClrSchemeLst" oxsd:cname="VariationColorSchemeList"
oxsd:cwrap="noTemplate">
  <xsd:sequence>
    <xsd:element name="variationClrScheme" oxsd:cname="variationClrScheme"
type="CT_VarClrScheme" minOccurs="4" maxOccurs="unbounded" />
```

```
  </xsd:sequence>
</xsd:complexType>
```


2.3.4.2.29 CT_VariationStyle

Target namespace: http://visThemeSchemaUri

Referenced by: CT_VariationStyleScheme

Specifies a style property of a style scheme list of a dynamic theme variant.

Attributes:

fillIdx: An xsd:unsignedInt ([XMLSCHEMA2] section 3.3.22) attribute that indirectly specifies the value of the properties of the QuickStyleFillMatrix Cell_Type element.

lineIdx: An xsd:unsignedInt ([XMLSCHEMA2] section 3.3.22) attribute that indirectly specifies the value of the properties of the QuickStyleLineMatrix Cell_Type element.

effectIdx: An xsd:unsignedInt ([XMLSCHEMA2] section 3.3.22) attribute that indirectly specifies the value of the properties of the QuickStyleEffectsMatrix Cell_Type element.

fontIdx: An xsd:unsignedInt ([XMLSCHEMA2] section 3.3.22) attribute indirectly specifies the value of the properties of the QuickStyleFontMatrix Cell_Type element.

Child Elements:

extLst: An **a:CT_OfficeArtExtensionList** ([ISO/IEC29500-1:2016] section 20.1.2.2.15) type which is unused and MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_VariationStyle" oxsd:cname="VariationStyle"
oxsd:cwrap="noTemplate">
    <xsd:sequence>
      <xsd:element name="extLst" oxsd:cname="ext" type="a:CT_OfficeArtExtensionList"
minOccurs="0" maxOccurs="1" oxsd:dataStructure="optional"/>
    </xsd:sequence>
    <xsd:attribute name="fillIdx" type="xsd:unsignedInt" use="required"/>
    <xsd:attribute name="lineIdx" type="xsd:unsignedInt" use="required"/>
    <xsd:attribute name="effectIdx" type="xsd:unsignedInt" use="required"/>
    <xsd:attribute name="fontIdx" type="xsd:unsignedInt" use="required"/>
</xsd:complexType>
```

2.3.4.2.30 CT_VariationStyleScheme

Target namespace: http://visThemeSchemaUri

Referenced by: CT_VariationStyleSchemeLst

Specifies a style scheme list of a dynamic theme variant.

Attributes:

embellishment: An xsd:unsignedInt ([XMLSCHEMA2] section 3.3.22) attribute that specifies embellishment information of a dynamic theme variant in a dynamic theme.

Child Elements:

VarStyle: A CT_VariationStyle type that specifies a style property of a style scheme list of a dynamic theme variant.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_VariationStyleScheme" oxsd:cname="VariationStyleScheme"
oxsd:cwrap="noTemplate">
    <xsd:sequence>
      <xsd:element name="varStyle" oxsd:cname="varStyle"  type="CT_VariationStyle"
minOccurs="4" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="embellishment" type="xsd:unsignedInt"/>
  </xsd:complexType>
```

2.3.4.2.31 CT_VariationStyleSchemeLst

Target namespace: http://visThemeSchemaUri

Referenced by: **Ext** element as detailed by the **CT_OfficeArtExtension** type specified in [ISO/IEC29500-1:2016] section 20.1.2.2.14.

Specifies four distinct style scheme lists of four distinct dynamic theme variants in a dynamic theme.

Child Elements:

VariationStyleScheme: A CT_VariationStyleScheme type that specifies a style scheme list of a dynamic theme variant.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_VariationStyleSchemeLst" oxsd:cname="VariationStyleSchemeList"
oxsd:cwrap="noTemplate">
  <xsd:sequence>
    <xsd:element name="variationStyleScheme" oxsd:cname="variationStyleScheme"
type="CT_VariationStyleScheme" minOccurs="4" maxOccurs="unbounded" />
  </xsd:sequence>
</xsd:complexType>
```

2.3.4.2.32 CustomMenusFile_Type

Target namespace: http://schemas.microsoft.com/office/visio/2011/1/core

Referenced by: DocumentSettings_Type

A complex type that is unused and MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CustomMenusFile_Type">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string"/>
  </xsd:simpleContent>
</xsd:complexType>
```

2.3.4.2.33 CustomToolbarsFile_Type